EyeVR - Low-Cost VR Eye-based Interaction

Florian Geiselhart
Ulm University
Ulm, 89079, Germany
florian.geiselhart@uni-ulm.de

Michael Rietzler
Ulm University
Ulm, 89079, Germany
michael.rietzler@uni-ulm.de

Enrico Rukzio
Ulm University
Ulm, 89079, Germany
enrico.rukzio@uni-ulm.de

Ubicomp/ISWC'16 Adjunct , September 12-16, 2016, Heidelberg, Germany
ACM 978-1-4503-4462-3/16/09.
http://dx.doi.org/10.1145/2968219.2971384

Abstract
The EyeVR system enables eye and gaze interactions in VR glasses at a price below $100, while providing sufficient performance for many typical VR use cases, like foveated rendering, gaming, or attention-based storytelling. It is based on off-the-shelf hardware like a Raspberry Pi, can be used in wireless, mobile settings and allows for a widespread use of gaze tracking in different applications through open interfaces. Besides standard gaze tracking, another focus of the system lies on exploring new forms of interactions besides only gaze direction, by providing additional details about the eye like pupil size or eyelid movement.

Author Keywords
Eye tracking, Gaze tracking, VR Interaction, Mobile VR.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous

Introduction
With the current rise of affordable, widespread VR headsets, new ways of interaction with the presented virtual content are again an ongoing research topic. Besides button- and gamepad-based interaction, which is often already integrated in nowadays devices, other interaction methods like gestural interaction e.g. using

head-mounted devices [2], touch interaction [5] or also eye-based interaction are of increasing interest for end-user products but also interaction research.

While e.g. gestural interaction can be already facilitated with a combination of cheap off-the-shelf components [2], eye-based interaction is still inaccessible to most end-users and lots of researchers due to the small range of available devices (e.g. [3]) and their high cost.

With EyeVR, we aim to close this gap by providing a system which allows to implement basic eye tracking tasks in VR at a cost of below $100.
We accomplish this by using off-the-shelf components and a rapid prototyping approach with the smallest possible number of parts.

System Architecture

The EyeVR system is based on the Samsung GearVR headset, using any compatible Samsung Smartphone as a display. The headset is extended by a camera and an infrared LED for monocular tracking, however binocular tracking is also possible by integrating the hardware for each eye respectively. The camera and LED are connected to a Raspberry Pi (v3) board, which is responsible for the image processing, and in turn provides raw gaze positions to the Smartphone via WiFi or Bluetooth. This offloading leaves all smartphone resources for the 3D scene rendering while ensuring sufficient resources for the tracking algorithm to run.

On the smartphone, an environment based on the Unity Engine is responsible for calibration, gaze-to-screen mapping and the rendering of the virtual scene.

Hardware

To capture a continuous image stream of the eye within the VR headset, a Raspberry Pi NoIR infrared camera module [1] mounted inside the headset is used. This camera is able to deliver infrared images at resolutions up to 1920x1080 and frame rates up to 90 fps. As the camera originally has not been designed for short range use, it is necessary to adjust the fixed lens focus to the very short eye-camera distance of around 2 cm. The camera is then fixed on a 3D-printed mounting bracket, which can be clipped around the the GearVR lens rings for easy attachment, positioning and removal.

The mounting bracket also houses the LED close to the camera lens, to create a corneal glint which can be used in turn for the tracking algorithm. The ideal position and viewing angle for the camera and LED, and thus the bracket parameters were derived via facial 3D scans and 3D modelling and simulation, as they are very sensitive and crucial for a good tracking performance.

The infrared LED is driven via the Pi GPIOs, using a DMA PWM approach to be able to control timing and brightness of the illumination.

Using a CSI-HDMI adapter board, the camera signal and LED control signal can be connected through a run of standard HDMI cable, which allows to place the Raspberry Pi and a USB power bank at a convenient location, e.g. as a belt pack or in a pocket. This enables mobile use and does not add any additional weight to the headset besides the lightweight camera assembly.

Figure 1: Hardware assembly inside the GearVR – the cables lead to a HDMI adapter board on the outside of the GearVR case.

Software

The software part of the EyeVR system consists of two parts - first, the image processing running in real-time on the Raspberry Pi, and second the Unity VR application on the smartphone which is placed into the Gear VR.

Tracking

The tracking algorithm on the Raspberry Pi is based on the dark pupil approach (e.g. described in [4]). It first detects the position of the corneal glint in the eye image, and then tries to find a dark ellipse close to the glint by using a contour finding method on the preprocessed imagery. This step is followed by a rating procedure to rank pupil candidates by their probability to be the correct one. An ellipse fitting step then estimates center and size of the detected pupil. The vector between the ellipse center and the corneal glint is then provided via Websocket connection or Bluetooth to the Unity Application on the smartphone. It is also possible to retrieve other parameters like blink probability and pupil diameter.

The algorithm is implemented on top of the OpenCV [6] library in Python, and is currently able to run at around 45 to 50 fps on the Raspberry Pi hardware, which allows for tracking of most of the voluntary eye movements when using VR glasses.

Calibration and Gaze Mapping

The second part of the software runs on the smartphone and is embedded in the Unity 3D VR application. It receives the raw data from the Pi tracker, and maps the raw glint-to-pupil vectors to screen positions. For this, currently an initial calibration step is necessary, which collects gaze data for a set of points in central and peripheral screen positions. After outliers have been removed and preprocessing has been applied, a mapping of gaze to screen coordinates is established by solving the resulting set of equations of a non-linear model using a singular value decomposition. The resulting screen-space gaze data can subsequently be used for any kind of VR application in Unity 3D.

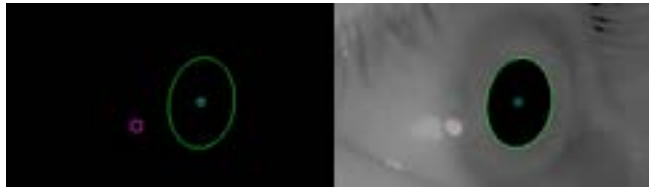

Figure 2: Output of the tracking algorithm (right) and source image with recognized glint and pupil as an overlay (left).

Applications and Performance

The system was built for research use in studies targeted towards eye-based interaction, and enables fast prototyping of eye tracking-enabled applications. However, it should be usable for a number of different use cases including simple forms of foveated rendering, or also estimation of cognitive states by measuring parameters like blink rate and pupil diameter. The VR headset environment provides very good conditions for such applications due to the great amount of control that can be exercised over lighting conditions and visual and/or auditory stimuli.

Conclusion and Future Work

We presented EyeVR, a working, low-cost approach to eye tracking in VR. It can be assembled out of available components for below $100 and is suitable for a number of VR applications. First tests showed the feasibility of the approach, already enabling basic tracking tasks. We currently work on improving the tracking algorithm for performance and greater robustness across different eye types and tracking situations, as well as binocular use for extended tracking possibilities. Furthermore, we plan to extend the calibration approach, which is currently based on previous calibration methods tailored for rectangular screen-shaped target areas, but should be modified and improved when being used in VR, where no rectangular screen space is present, but different virtual distances can occur.

We also plan to conduct studies to measure the performance and technical aspects of the tracker as well as the user acceptance. Besides this, different possible applications are currently under evaluation and will be further investigated in dedicated studies.

References

1. Raspberry Pi NoIR camera website
Retrieved July 24, 2016 from
https://www.raspberrypi.org/products/pi-noir-camera-v2/
2. Leap Motion VR setup
Retrieved July 24, 2016 from
https://developer.leapmotion.com/vr-setup
3. SMI AR/VR eye tracking kits
Retrieved July 24, 2016 from
http://www.smivision.com/oem-eye-tracking/
4. Morimoto, Koons, Amir, Flickner. 2000.
Pupil detection and tracking using multiple light sources.
Image and vision computing 18.4 (2000): 331-335.
http://dx.doi.org/10.1016/S0262-8856(99)000530
5. Gugenheimer, Dobbelstein, Winkler, Haas, Rukzio. 2016. FaceTouch: Touch Interaction for Mobile Virtual Reality.
Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (2016): 3679-3682
http://dx.doi.org/10.1145/2851581.2890242
6. OpenCV 3.0 library
Retrieved July 24, 2016 from
http://opencv.org/