An Affordable Solution for Binocular Eye Tracking and Calibration in Head-mounted Displays

Michael Stengel[1], Steve Grogorick[1],
Martin Eisemann[2], Elmar Eisemann[3], Marcus Magnor[1]
[1]TU Braunschweig, Germany [2]TH Köln, Germany [3]TU Delft, Netherlands
{stengel,grogorick,magnor}@cg.cs.tu-bs.de, martin.eisemann@fh-koeln.de, e.eisemann@tudelft.nl *

ABSTRACT

Immersion is the ultimate goal of head-mounted displays (HMD) for Virtual Reality (VR) in order to produce a convincing user experience. Two important aspects in this context are motion sickness, often due to imprecise calibration, and the integration of a reliable eye tracking. We propose an affordable hard- and software solution for drift-free eye-tracking and user-friendly lens calibration within an HMD. The use of dichroic mirrors leads to a lean design that provides the full field-of-view (FOV) while using commodity cameras for eye tracking. Our prototype supports personalizable lens positioning to accommodate for different interocular distances. On the software side, a model-based calibration procedure adjusts the eye tracking system and gaze estimation to varying lens positions. Challenges such as partial occlusions due to the lens holders and eye lids are handled by a novel robust monocular pupil-tracking approach. We present four applications of our work: Gaze map estimation, foveated rendering for depth of field, gaze-contingent level-of-detail, and gaze control of virtual avatars.

Categories and Subject Descriptors

C.3 [**Computer Graphics**]: Special-Purpose and Application-based Systems - Real-time and embedded systems; I.3.1 [**Computer Graphics**]: Hardware Architecture - Input devices; I.3.7 [**Computer Graphics**]: 3D Graphics and Realism - Virtual Reality; I.3.8 [**Computer Graphics**]: 3D Graphics and Realism - Applications

Keywords

eye tracking; gaze; wearable; virtual reality; head-mounted display; mobile

1. INTRODUCTION

Virtual Reality (VR) has become a well-established field in research and industrial applications, e.g., for simulations, scientific visualization, or gaming. Previously, high hardware costs prevented a wide-spread application and development. But recent advances in the mobile device market, lead to high-quality, low-cost virtual reality hardware (*Oculus Rift*, *HTC Vive*, *Sony PS4 HMD*, etc.). These low-weight, low-latency head-mounted displays (HMDs), in combination with a wide field of view (FOV), enable a never before experienced *immersion* and *presence* within a virtual environment. Future developments of HMDs will include even higher resolution displays, higher refresh rates, and wider FOVs [8].

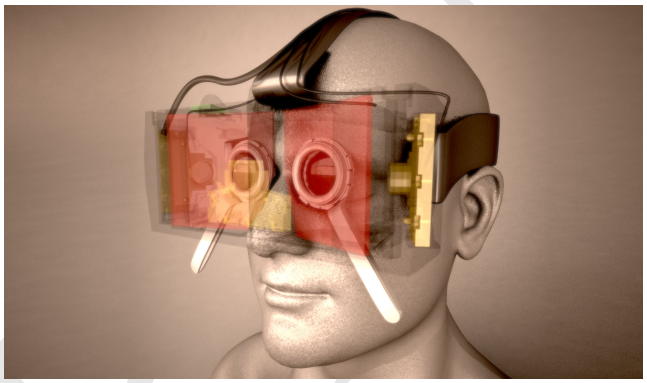

Figure 1: **Prototype visualization**. A rendering of our self-contained eye tracking head-mounted display. Based on a system of dichroic mirrors (red), infrared light illuminating lense holders (white) and tracking cameras (yellow) the device captures the user's eye motion for binocular eye-tracking while he is fully immersed in the virtual world.

Commodity HMDs mostly use fixed hardware setups. However, preconfigured HMDs are often difficult to parameterize for individuals, e.g., to account for differing interocular distances; both, in horizontal and vertical direction (previously often ignored and known as Hypertropia [17]). Further, existing software calibration is often unsatisfactory and cumbersome with current HMDs. This limitation can lead to non-frontal relative positioning of the eye and non-converging lenses inside the HMD, resulting in reduced perceived sharpness, and an increased likelihood of motion sickness and headaches for the user.

The wide adoption of VR equipment makes it crucial to investigate methods to simplify calibration and to improve the experience for each user. Here, analyzing user behavior in virtual environments can deliver many insights: What is drawing attention? What emotional response results from certain content?

For a desktop setup congeneric findings are usually investigated involving an eye tracker (measuring pupil size for emotions or focus points of interests on the screen). Unfortunately, when using an HMD setup, the integration of eye tracking is not straightforward and existing solutions are not convenient for commodity HMDs.

*Video and further paper details available under:
http://graphics.tu-bs.de/publications/stengel2015acmmm/

©2015 ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record will be published in Proceedings of ACM Multimedia 2015, http://dx.doi.org/10.1145/2733373.2806265.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

PREPRINT

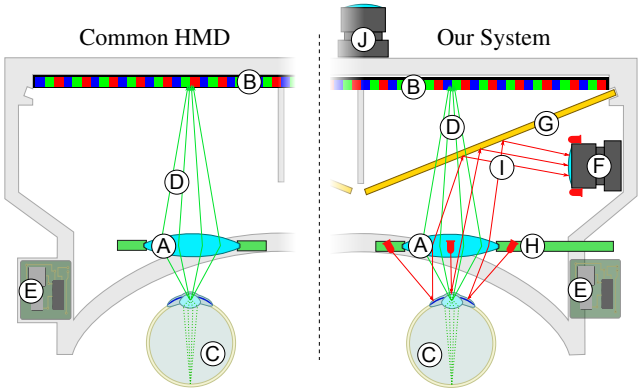

Figure 2: **HMD design comparison.** Common HMDs setup (left part): converging lens (A), display (B), eye ball (C), visible light (D), orientation tracker (E); Our system adds (right part): eye tracking cameras (F), dichroic mirrors (G), lens controller with infrared LEDs (H), infrared light (I), positional tracking camera (J).

Stationary solutions for eye tracking are state-of-the-art with regard to tracking quality and mostly applied to estimate scan paths (fixations and saccades) [26, 36]. The user's head is locked in a position using a rigid positioner and a camera records the eye. While the systems are very accurate at a high tracking sample rate, the fixed viewing position is not an option for immersive VR and even small head movements lead to drifts if recalibration is not frequently performed.

Mobile eye-tracking solutions overcome the motion restrictions. In this case, an integration into a headgear or special-glasses frame enables free head movement (e.g., SMI Eye Tracking Glasses, Arrington Research 3DViewPoint^{TM}, Biopac Systems, Inc. HMD). However, due to the smaller form factor it is significantly more ambitious to integrate such a solution into an HMD: The tracking relies on a camera, whose position is constrained by the HMD lenses and lens holders, which would block the view partially. Hence, a point right below the eyes is chosen, where precision is unfortunately non-uniform. An alternate more-frontal placement inadvertently reduces the FOV, which is often not an option because the feeling of immersion only starts at a horizontal FOV of 80°and increases quickly until 110°[8, 14]. Such an eye tracking procedure is further complicated since typical corneal-reflection-based eye-tracking algorithms [23] are not applicable, as they would produce disturbing reflections on the lenses.

Our work addresses these limitations and works towards gaining more insights into the VR experience. We propose an affordable, drift-free, and binocular eye-tracking solution, which is usable within the limited space of current HMD hardware designs without FOV reduction (Fig. 1). Throughout this paper, we will show how to overcome the challenges involved in designing such a VR system and solve several other issues, for instance calibration and adaptation to the user. Specifically, our contributions are:

1. a personalizable lens positioning system (horizontal and vertical) for HMDs and an integration of an unobtrusive camera setup for eye tracking in a lens-based HMD based on a specialized infra-red lighting (Sec. 3);

2. a model-based gaze estimation algorithm and calibration procedure to adjust the system to the user (Sec. 4);

3. a robust monocular pupil-tracking algorithm, which can deal with partial eye occlusions due to the lens holders and eye lid (Sec. 5);

We present applications showing the potential of our (binocular) eye tracking HMD, which are foveated rendering for depth of field, gaze-contingent level-of-detail, gaze map creation and realistic gaze control of virtual characters. In general, these applications illustrate the ability of our system to also perform psychophysical experiments and to extend the experience in immersive environments (Sec. 6). To validate our proposed system, we performed an objective comparison with a state-of-the-art pupil-tracking algorithm for near-field eye-trackers [31] and conducted a user evaluation (Sec. 7). We discuss the limitations of the current setup and give an outlook on future work (Sec. 8) before concluding this paper (Sec. 9).

2. RELATED WORK

Head-mounted Eye Trackers The success of the Oculus-Rift HMD led to a renewed interest in VR for the consumer market. The most-evolved HMDs in this low-cost sector, Crystal Cove (Oculus VR) and HTC Vive, offer a display resolution of at least Full-HD as well as positional and rotational tracking. Eye tracking is a natural next step and gained much attention in the research and development sector (e.g., FOVE Inc., Arrington Research, ASL Eye-Trac 6, SR Research, or Senso Motoric Instruments (SMI)). Even though first attempts have been undertaken in the year 2000 [20], current prototypes are still far from being consumer-ready with prices up to 15,000$ (SMI's eye tracker in the Oculus Rift case). One major cost factor are the miniature cameras and specialized digital processors for tracking at high speed. While the interior design of these Eye-tracking HMDs (ETHMD) is mostly kept secret, the comparatively low vertical FOV could suggest that the camera is placed inside the user's FOV occluding parts of the display. Our eye-tracking HMD setup has several benefits. It is a low-cost solution (approximately 450$), which offers the full FOV of current state-of-the-art HMDs.

Closest to our design is the EyeSeeCam [37]. This wearable eye tracker is used to align the focus of an external camera and the user in real-time for medical applications, surgery, or behavioral sciences. We similarly use dichroic mirrors to reflect infrared light from the eyes back to the cameras located outside the FOV. The custom-built EyeSeeCam can rely on traditional eye-tracking algorithms, but is also more expensive. We propose a customization, but it implies additional challenges to be solved, such as partial occlusions by the lens holders and view distortions by the lenses.

Eye-Tracking Algorithms Eye-tracking algorithms have a long research history and are often optimized for specialized setups which vary greatly in their design. A more general survey on eye tracking, including the employed eye models can be found in [23], while methods to evaluate eye-tracking quality are presented in [27]. Here, we will focus on the most-related work.

An essential step for feature-based eye tracking is the pupil detection. It forms the darkest part of the eye if illuminated from an off-axis view, and the brightest part if illuminated from a near-camera-axis view. Consequently, it is often well-separable from the surrounding iris. Most techniques rely on edge or contour detection of the pupil followed by an ellipse fitting. The main challenge is to deal with glints and reflections [30, 16], blinks [15], or noise [31], especially for near-eye devices, as e.g. Google Glasses [15, 11]. The gaze is then often estimated using the Purkinje effect (a bright glint appearing in the recorded infrared image resulting from the reflection of a spotlight LED used for illumination) in conjunction with camera and LED positions [18]. A taxonomy of the different approaches is given in [39].

In an HMD the Purkinje effect cannot be directly applied due to the converging lenses. LED illumination in front of the lenses

PREPRINT

(camera side) results in visible reflections from the lenses themselves, while a placement behind the lenses (closer to the eye) has the drawback that glints will not always be visible for the targeted wide FOV. We will show how to overcome this challenge.

Appearance-based methods do not extract features (pupil or glints) but use the complete input image to estimate the observed screen position. Several approaches built upon this idea in form of multilayer networks [10], Gaussian processes [40] or manifold learning [38]. While flexible (requiring only a calibration step), these techniques are often computationally costly and less applicable for VR scenarios, where high tracking rates are required. We make use of a novel approach combining image features with an underlying simulation model of our HMD; we detect the pupil in the recorded views, calibrate our HMD model, and derive the observed screen pixel position using a physical eye model.

3. EYE-TRACKING HMD

In the following, we describe our low-cost low-weight and personalizable design for immersive VR with unobtrusive eye tracking. After a general overview, we describe the details of each of the HMD's main components (Sec. 3.1).

3.1 Device Construction

General The most important elements are visualized in Fig. 2 and the working prototype in Fig. 3a–b. Our basic setup resembles a classic HMD with converging lenses (A) to focus the view on the display (B). The difference lies in infrared cameras at the outer boundary of the body case (F), dichroic mirros (G) and a circular LED-light array along the adjustable lens holders (H) to illuminate the eye (C). Reflected infrared light passes through the converging lenses and is reflected towards the cameras via two tilted dichroic mirrors (G). Display light (D) passes unhindered towards lenses and eye. An additional front camera (J) is used for markerless positional tracking and an integrated Inertial Measurement Unit (E) for orientation [21]. The electronic components are wired to a single harness connected to an external box with the display controller, an Arduino for orientation tracking, and the LED power supply [3].

Body Case The body case encapsulates all internal components (Fig. 2). A central barrier with a gap for the nose divides the display (C) into two disjoint symmetric parts, one for each eye. The case is firmly closed and tightened with foamed material to avoid exterior stray light and covered with comfortable tissue, except at the nose tip to enable normal breathing. The dimensions of the body were determined by fitting it to several 3D head scans (Fig. 3c).

Display We integrated a 5.6" LCD (1280×800 pixel resolution) with a refresh rate of 60 Hz. As indicated before, display controller and display are separated, which reduces the HMD weight.

Converging Lenses We use converging lenses of an Oculus Rift (DK1) to increase the perceived field of view [6]. Hereby, we also maintain compatibility to the Oculus Rift. Our prototype offers a horizontal field of view of 86°per eye. We provide dedicated controllers to adjust the position of the lenses in both horizontal and vertical direction for optimal lens placement (Fig. 1). Compared to a traditional HMD with interchangeable lens cups, our design makes more flexible and precise adjustments for varying head anatomy possible. For calibration, a circular IR reflecting ring is located on the backside of the lens holders.

Dichroic Mirrors We use two dichroic flat surfaces (also known as *hot mirrors*), which reflect light at wavelengths longer than 730 nm (infrared), while short wavelengths (< 720 nm) are entirely transmitted. They redirect infrared light reflected by the illuminated eyes towards the integrated cameras, which allows us to track the gaze without obscuring the field of view of the user.

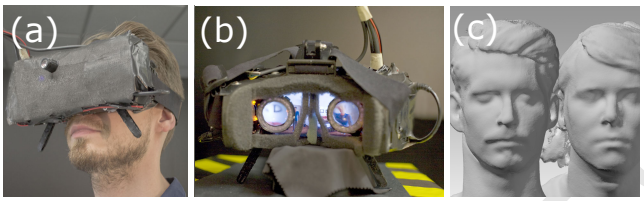

Figure 3: **HMD design and assembly**. User wearing the eye tracking HMD (a), working prototype (b), 3d scans of different human heads used for case dimensioning (c).

Figure 4: **Eye illuminating lense holder.** 3d printed lense holder with manufactured circuit board (a), working infrared SMD-LED array (b), illumination units within the HMD (c).

The dichroic mirrors have a size of $80 \times 80 \times 2$ mm with central cutouts for the nose and an inclination angle of 19.5°along the vertical axis. The angle is a tradeoff between space and optimal view on the eye (45°inclination). Higher inclination angles increase the necessary screen distance and, thus, screen size and weight. Smaller inclination angles lead to partly occluded views on the eye, which need to be dealt with during the pupil tracking.

Infrared Illumination Unit Twenty-five infrared LEDs mounted on a ring circuit lead to a uniform eye illumination from all directions (Fig. 4). The circuit has an inner diameter of 37 mm and width of 1.5 mm to minimize the lens controller size. The LEDs radiate with a wavelength of 830 nm over a wide angle of 140°. This infrared light enhances the pupil's contour in comparison to the contour of the limbus, but is outside the visible spectrum, thus, invisible to the user. We ensure safety of the user with regard to the impact on the user's exposure to the involved infrared radiation. An analysis can be found in the supplemental material.

Eye Tracking Cameras For binocular eye tracking, we integrated two low-cost cameras focusing at the distance of the user's eyes and having a fixed diagonal field of view of 56°. We exchanged their infrared filters with a long-pass filter blocking all but infrared light, in form of a single layer of a raw film negative. The cameras have fixed positions in the HMD (Fig. 2 (F)) and record at a 640×480 pixel resolution in grayscale at 75 Hz with a delay of 13 ms due to the internal image processor. This sampling rate suffices to track *fixations* and *smooth pursuit eye movements*.

Head Tracking For the viewpoint estimation in a virtual environment the rotational and translational component of the HMD are required as well. To this extent, we integrated an orientation sensor into the HMD and perform positional tracking via a head-mounted front camera. This combined setup is inexpensive and enables the required 6-degrees-of-freedom head tracking with sufficiently high precision and low latency.

We include an inertial measurement unit (IMU) holding multiple sensors connected to an Arduino microcontroller board to track the orientation (Yaw-Pitch-Roll). The IMU consists of an accelerometer, a gyro sensor and a digital motion processor (DMP). We set the update rate to 200 Hz to avoid a noticeable delay when moving the head and to reduce motion sickness.

The DMP supports automatic self-calibration and the angular drift of the IMU is less than 1° per minute, which is sufficient for

PREPRINT

longer usage. The positional tracking of the IMU could suffer from an integration error over time, resulting in an accumulated drift. However, over short periods of time, the IMU delivers sufficiently precise data. Our tracking solution combines the low-latency IMU output with markerless camera tracking, which results in a robust low-latency positional tracking with good precision. The pose estimation of the HMD front camera in world space is based on SLAM-feature tracking, implemented in the Metaio SDK [21, 2]. Our pose estimation proceeds as follows: The world frame is oriented and positioned automatically after a few seconds of feature initialization. Features are then detected and refined adaptively over time during tracking. Since world scale cannot be estimated from the tracker, it is automatically adjusted in the initialization phase, such that the camera-tracker results share the magnitude of the velocity measured by the IMU. This positional tracking takes ≈ 23 ms in our setup (13 ms for frame transmission, 10 ms for pose estimation).

4. CALIBRATION

This section describes calibration and reconstruction procedures for the different HMD components (Sec. 4.1) and the user-specific calibration (Sec. 4.2). Both are required for precise eye tracking, gaze estimation, and personalized adjustments. It is an important step in adapting the device to the user, which, ultimately, leads to a better VR experience. We describe the setup for one eye, the second eye is handled equivalently. The eye tracking implementation will be described in Sec.5.

4.1 HMD Calibration

To avoid motion sickness and create a convincing 3D impression, we require precise knowledge about each component in our HMD projection chain, meaning the relative position and orientation of the eye-tracking camera, the dichroic mirror, the lens and lens holder, as well as the refractive properties of the converging lens, and the intrinsic parameters of the eye-tracking camera. As a reference point $\mathbf{o}_H$ for all components of the HMD, we use the horizontal center of the HMD's front-most point.

Eye-Tracking Camera Calibration. We estimate both the intrinsic and extrinsic parameters for the eye-tracking camera. Intrinsics are derived via the technique by Bouguet [13]. Providing image resolution and sensor size is sufficient to transform a recording of a checkerboard or circle pattern of known size on a flat surface into focal length, principal point, as well as radial and tangential lens distortion.

The extrinsic camera parameters are derived during production as follows. Before the dichroic mirror is inserted into the body case, we cover the screen with a checkerboard calibration pattern, which is carefully adjusted to align with the edges of the body case. The eye tracking camera records the pattern and the extrinsic parameters are derived in relation to the pattern. We use the same CAD model, which we used to print the body case, and transform the extrinsic camera parameters into $\mathbf{o}_H$, the coordinate system of the HMD [24]. This virtual/real-world relationship will be exploited for the calibration. For validation, we compare the captured image with a rendered version of the checkerboard using the derived camera parameters. The reprojection error is less than 3 pixels and would be further reduced in an industrial production setting.

Mirror Calibration. After the camera has been calibrated the dichroic mirror is inserted and calibrated. We cover the mirror with a carefully aligned calibration pattern, to later match it to the CAD model, and capture it from the eye-tracking cameras. Performing

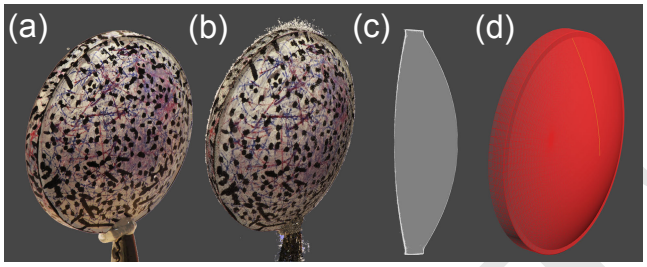

Figure 5: **Lens reconstruction**. Converging lens with artificial surface features (a), reconstructed 3d point cloud (b), derived lens profile (c), smooth reconstructed model (d).

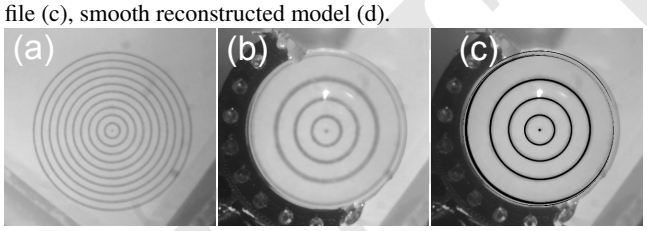

Figure 6: **Refractive index estimation**. Calibration object (a), ground truth refraction through the lens (b), rerendered calibration object (c) rendered on top of (b).

the same calibration procedure as for the cameras this gives us the camera parameters in relation to the mirror position, and vice versa. We then transform this relative mirror position into $\mathbf{o}_H$. Again, we validate the correctness of the derived parameters by rendering the checkerboard and comparing it with the captured image. In our prototype, the rotation angles of the mirrors were $\sim 18.9°$ and $\sim 19.5°$ for the left and right mirror, respectively. The slight asymmetry was due to a fabrication imperfection when printing the HMD.

Lens Reconstruction. We require an accurate geometric model of the aspheric lens as well as the index of refraction (IOR) to support the user calibration later on. In our case, details about the used optics have not been available and accordingly had to be reconstructed. As this is the situation for most lens models in HMDs, we describe our lens reconstruction approach in the following.

Lens Geometry To avoid a complicated reconstruction of a transparent surface, we artificially colorize the lens with ink and create a set of discriminative features (Fig. 5a). We then reconstruct a lens-surface point cloud based on different input views [1], which we capture at high quality and resolution using a DSLR camera (Fig. 5b). As a point cloud may contain holes, we fit a parametric lens model (Fig. 5c) as follows. We assume a disc-like and radially-symmetric shape. The mean positional vector μ of the point cloud and the eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ provide a convenient coordinate space for the lens reconstruction, as μ is equal to the center of the lens and together with $\mathbf{e}_3$ describes the rotation axis $\mathbf{r} = \mu + t\mathbf{e}_3$. Because of the symmetry assumption, we only need to derive the 2D profile (Fig. 5c). It can be conveniently described by two 2nd order polynomials for the front and back curvature. We rotate each point of the point cloud around $\mathbf{r}$ onto the plane centered at μ and spanned by $\mathbf{e}_1, \mathbf{e}_3$. We then fit two 2nd order polynomials to the point cloud, one for the front facing points and one for the backfacing points [33]. This approach also increases robustness as the symmetry assumption leads to a better use of the redundancy in the point cloud. The lens is then reconstructed from this parameterized profile (Fig. 5d).

Index of Refraction Since the lens' index of refraction (IOR) is wavelength-dependent, we estimate it for infrared and the visi-

PREPRINT

ble spectrum to incorporate both light paths in our HMD (Fig. 2). The following procedure is the same for both, only the recording camera is exchanged. We apply an analysis-by-synthesis approach based on the lens' geometric properties. First, we record a front view of a circular calibration pattern (Fig. 6a) having an outer diameter of 50mm at a known distance and calibrate the cameras as before. After adding the lens between camera and pattern we take several images at different known distances between pattern and lens (Fig. 6b). We then optimize the IOR by comparing synthetically rendered scenes of the lens and calibration pattern (Fig. 6c) using the physically-correct and wavelength-dependent Maxwell Renderer [5] to the recorded camera images. We estimated an IOR $N_I = 1.472$ for the wavelength $\lambda = 950nm$ and $N_V = 1.515$ for $\lambda = 560nm$ which are typical values for materials like Acrylite, Lucite or Plexiglass.

4.2 User Calibration

Most components of our system can be calibrated at production time (Sec. 4.1). User-specific components, such as the lens holder position, interpupillary distance, and eye-to-lens distance need to be estimated for every user separately. These are important for a natural 3D impression and meaningful eye-tracking results, as they are essential to predict the virtual viewpoint, which can otherwise only be roughly estimated. The components of the gaze simulation model being calibrated are visualized in Fig. 7a. First, the user adjusts the lenses parallel to the screen, to have a frontal view when looking straight. Next, the lens distance is adjusted until the screen appears sharp.

Lens-Holder Localization. To detect the lens-holder position, and, hereby, the lens' position, we use the white IR reflecting ring on the backside of the lens holder (Fig. 7a). Additionally added infrared LEDs are located around the eye tracking camera solely for illuminating the ring (Fig. 2 (F), red LEDs at the camera). When turning off the screen and the interior LED ring, the lens holder can be detected by thresholding the image captured by the eye-tracking camera. We then derive its center and eccentricity [22]. We compute the 3D position and orientation of this ring again via an analysis-by-synthesis procedure; we render a model of the ring and iteratively optimize its position and rotation via a gradient-descent approach based on the difference between the ellipse centers, size and eccentricity, which proved fast and accurate.

Eye Calibration. Next, we estimate the eye's distance to the lens. The main problem is that a view of the eye does not provide useful information regarding scale, as eye sizes differ. Further, the view might be distorted in complex ways by the converging lenses. Analysis-by-synthesis can again help us in this situation. The LED ring in the lens holders produces a characteristic reflection on an eye, also known as glint (Fig. 7b–c). This reflection can be used to determine the distance between lens and eye. Nonetheless, to make this step possible we need a physically-plausible eye model.

Eye Model The eye ball of a healthy adult human has a quite consistent shape [9]. The main part can be modeled as a sphere rotating around its center with a diameter of 24mm and only few individual deviations (Gaussian distribution with a standard deviation of $\pm$1mm). The cornea forms an additional spherical surface above the sclera with a smaller radius of 7.8mm. The direct light reflected from the sclera produces the most prominent glints. We set the IOR of the cornea to $N_D = 1.2$ and the eye fluids to $N_D = 1.276$ [9].

Eye Registration We extract the glints using a simple threshold $t_G = 0.9$. When a user looks along the optical axis of the eye tracking camera (taking the reflection from the dichroic mirror into account) the glints form an almost perfect circle on the sclera, otherwise this circle is distorted. The shape of the glints can, thus, be used in a feedback loop to guide the user's view towards the optical axis (Fig. 7b–c). To this extent, the user is asked to focus on a marker on the screen. The glints from the resulting image are extracted and an ellipse is fitted to it [22]. The marker is then moved and the ellipse is evaluated again. The movement of the marker is given by $\alpha(c_g - c_p)$ where c_g is the glint-ellipse center in pixel coordinates and c_p the estimated pupil-center position. The process is computationally cheap and α can be small, which lets the marker smoothly move over the screen until the algorithm converges.

We then derive the eye-lens distance and the absolute 3d position of the eye based on the eye model. In practice, we rendered the characteristic positions, where the glints are as circular as possible, for different eye distances and positions and record the diameter and center of the fitted ellipse. The result is a Look-Up table, which allows us to calibrate for the eye position quickly.

Gaze Calibration. Finally, for the gaze estimation, we need a mapping from pupil positions in the eye-tracking camera to screen positions. We rely on our virtual HMD model configured with the derived calibration values and estimated eye position. Here, we compute the light path from a pixel, representing a detected pupil center, of the eye-tracking image over the dichroic mirrors, through the lens towards the eye. By construction, this ray has to cross the eye at the pupil center (Fig. 2 red light paths). We can, thus, map eye-tracking camera pixels to an eye rotation. Similarly, the eye rotation can be used to determine a screen position by computing the light path from the eye through the lenses onto the screen (Fig. 2 green light paths). This mapping is precomputed for approximately 1300 virtual eye rotations per eye covering the full motion range of the human eye (Fig. 8k, black and red dots). Using barycentric interpolation, we can then map each potential pupil position c_p in the eye-tracking camera (Fig. 8k, green dot), to a view vector $\vec{v}$ and a pixel position on the display c_s.

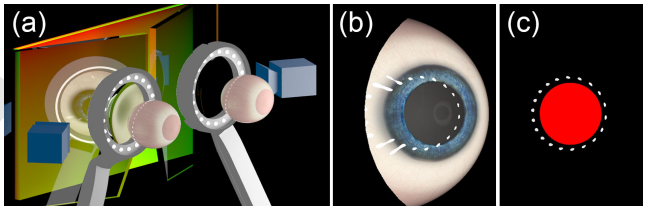

Figure 7: **Simulated gaze model**. Virtual gaze setup (a), realistic synthetic eye (b), glints and pupil mask for characteristic gaze (c).

5. MONOCULAR PUPIL TRACKING

Our gaze estimation (Sec. 4) relies on the current pupil position in the eye-camera image. The pupil extraction is described in this section, for which noise, (partial) occlusions by the eye lid or lashes, and dust or smears on the lens or mirror need to be handled.

Since our off-axis illumination units result in a *dark pupil*, we detect low intensities, which differ significantly from the high amount of reflected infrared light from the sclera and limbus. The limbus only absorbs more light in the visible light spectrum.

For the pupil tracking (Alg. 1), we rely on the grayscale-image $\mathbf{I}$ normalized to $[0,1]$ and a binary mask $\mathbf{M}_L$ indicating pixels belonging to the lens, which is obtained during the calibration step. For real-time performance and robustness, we first determine whether the eye is closed, open or halfway closed. Each configuration is dealt with separately (Alg. 1).

PREPRINT

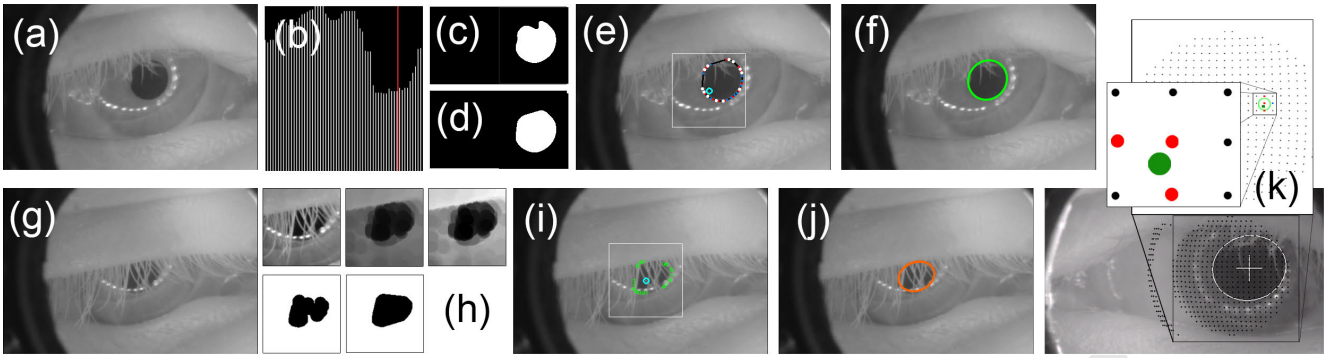

Figure 8: **Pupil detection pipeline.** Top row: Visible pupil case, Alg. 4. Captured image (a), histogram for threshold estimation (b), pupil binarization (c), pupil closing (d), contour filtering (e), pupil ellipse fitting (f). Bottom row: Partially occluded pupil case, Alg. 5. Captured image (g), pupil filtering and binarization (h), contour point filtering (i), pupil ellipse fitting (j). Barycentric interpolation for pupil-to-screen mapping (green: current pupil position, black: precomputed positions, red: closest samples used for interpolation) (k).

Algorithm 1 Pupil Tracking $(\mathbf{I}, \mathbf{M}_L)$

$\widetilde{p} \leftarrow$ Approximate Pupil Position $(\mathbf{I}, \mathbf{M}_L)$ ▷ Alg. 2
if $\mathbf{I}(\widetilde{p}) > t_{visibility}$ **then**
  Eye is closed
else
  $\theta \leftarrow$ Compute Pupil Occlusion $(\mathbf{I}, \mathbf{M}_L, \widetilde{p})$ ▷ Alg. 3
  if $\theta < t_{occlude}$ **then**
    $\{p, e_x, e_y, \phi\} \leftarrow$ Detect Visible Pupil $(\mathbf{I}, \mathbf{M}_L)$ ▷ Alg. 4
  else
    $\{p, e_x, e_y, \phi\} \leftarrow$ Detect Occluded Pupil $(\mathbf{I}, \mathbf{M}_L)$ ▷ Alg. 5
  end if
end if
return $\{p, e_x, e_y, \phi\}$

Approximate Pupil Position To make a fast guess of whether the eye is closed or not we approximately locate the pupil position as follows (Alg. 2):

Algorithm 2 Approximate Pupil Position $(\mathbf{I}, \mathbf{M}_L)$

$p_{cum} \leftarrow (0,0) \quad w_{cum} \leftarrow 0$
for $p \in \mathbf{M}_L$ **do**
  $w \leftarrow (1 - \mathbf{I}(p))^{\gamma}$
  $p_{cum} \leftarrow p_{cum} + p \cdot w$
  $w_{cum} \leftarrow w_{cum} + w$
end for
return $\widetilde{p} \leftarrow p_{cum} / w_{cum}$

We accumulate a weighted average of all pixel positions p within the lens mask $\mathbf{M}_L$. Each pixel p contributes with a weight w determined by $(1 - \mathbf{I}(p))^{\gamma}$ with $\gamma = 10$. Hence, darker pixels (higher likelihood to be the pupil) will contribute more. The weighted-average position is our initial pupil-position guess $\widetilde{p}$.

Occlusion Estimation If the intensity in a 70×70 pixels wide window around the initial pupil position is above the threshold $t_{visibility} = 0.4$, the eye is regarded as being closed. If the eye is not completely closed, we further refine our strategy by classifying it as either completely visible or partially occluded. The amount of occlusion is defined by two measures m_1 and m_2 (Alg. 3). While not being sufficient on their own the combination is significantly more robust. The first m_1 estimates the presence of eye lashes. The second m_2 tries to detect the amount of non-pupil pixels. Based on their combination, the eye is declared visible or partially visible and the corresponding detection algorithms are run.

To compute m_1 and m_2, we first remove glints by inpainting all pixels with a higher intensity than $t_G = 0.9$ [12]. These appear especially in the 1.5 – 2 mm transition zone of the curvature of the sclera and the curvature of the corneal surface that forms an external and internal surface groove (*scleral sulcus*) [9]. For more conservative results we slightly dilate the inpainted area $\mathbf{M}_{Glints}$.

We then detect eye lashes occluding the pupil in the resulting image $\mathbf{I}_{NoGlints}$. We restrict our computations to a small area $\mathbf{M}_{ROI}$ of radius $r = 35$ pixels around $\widetilde{p}$. Then a morphological opening filter is applied (minimum before maximum filter) to $\mathbf{I}_{NoGlints}$ with a kernel size $k_{MinMax} = 13$, removing finer structures, such as eye lashes. The first term m_1 is then defined as the sum of absolute intensity values of the difference image $\mathbf{I}_{\Delta} = |\mathbf{I}_{MinMax} - \mathbf{I}_{NoGlints}|$.

The second term aims at estimating the number of non-pupil pixels, which are brighter. To this extent, the gray value range $[0.4, 0.7]$ in $\mathbf{I}_{NoGlints}$ is linearly mapped to the range $[0,1]$. Other values are clamped accordingly. The second term m_2 is then defined as the sum of the resulting intensities inside $\mathbf{M}_{ROI}$. Both terms are combined into the final occlusion score $\theta = 0.5 \cdot (1/3 m_1 + 2/3 m_2)$. If $\theta < t_{occlude} = 0.3$ the eye is considered visible otherwise as partially occluded. The corresponding detection algorithm is applied.

Algorithm 3 Compute Pupil Occlusion $(\mathbf{I}, \mathbf{M}_L, \widetilde{p})$

$\mathbf{M}_{Glints} \leftarrow \{p \in \mathbf{M}_L \mid \mathbf{I}(p) > t_G\}$ ▷ Glint Mask
$\mathbf{M}_{Glints} \leftarrow F_{\text{Dilate}}(\mathbf{M}_{Glints})$
$\mathbf{I}_{NoGlints} \leftarrow F_{\text{Inpainting}}(\mathbf{I}, \mathbf{M}_{Glints})$ ▷ Glints removed
$\mathbf{I}_{MinMax} \leftarrow F_{\text{Min}}(\mathbf{I}_{NoGlints}, k_{MinMax})$ ▷ Eye lashes removed
$\mathbf{I}_{MinMax} \leftarrow F_{\text{Max}}(\mathbf{I}_{EyeLashes}, k_{MinMax})$
$\mathbf{I}_{\Delta} \leftarrow |\mathbf{I}_{MinMax} - \mathbf{I}_{NoGlints}|$
$\mathbf{M}_{ROI} \leftarrow F_{\text{circMask}}(\mathbf{I}, \widetilde{p}, r_1) \cap \mathbf{M}_L$
$\mathbf{I}_{\Delta} \leftarrow \text{Normalize}(\mathbf{I}_{\Delta} \cap \mathbf{M}_{ROI})$
$m_1 \leftarrow \sum_{p \in \mathbf{M}_{ROI}} \mathbf{I}_{\Delta}(p) / |\mathbf{M}_{ROI}|$ ▷ First Metric
$\mathbf{I}_{EyeLashes} \leftarrow F_{\text{(Tonal Correction)}}(\mathbf{I}_{NoGlints}, [0.4, 0.7])$
$\mathbf{M}_{ROI} \leftarrow F_{\text{circMask}}(\mathbf{I}, \widetilde{p}, r_2) \cap \mathbf{M}_L$
$m_2 \leftarrow \sum_{p \in \mathbf{M}_{ROI}} \mathbf{I}_{EyeLashes}(p) / |\mathbf{M}_{ROI}|$ ▷ Second Metric
$\theta \leftarrow (m_1 \cdot w_1 + m_2 \cdot w_2) \cdot 0.5$ ▷ Combined Metric
return θ

Visible Pupil We will now describe how to localize a visible or moderately-occluded pupil (Alg. 4). We build upon the observation that pupil pixels in comparison to its surrounding are well separated in an image histogram (Fig. 8a–b). We thus compute a histogram

PREPRINT

Algorithm 4 Detect visible pupil $(\mathbf{I}, \mathbf{M}_L)$

$\mathbf{I}_{BP} \leftarrow 1 - \mathbf{I}$ ▷ Inverts to bright pupil image
$\mathbf{H} \leftarrow \text{Histogram}(\mathbf{I}_{BP}, \mathbf{M}_L)$
$\mathbf{H} \leftarrow F_{\text{Median}}(\mathbf{H}, k_{HistMedian})$
$h \leftarrow \text{findGrayvalueOfBrightestLocalMinimum}(\mathbf{H})$
$\mathbf{M}_{PupilSeg} \leftarrow \{p \in \Omega \mid \mathbf{I}_{BP}(p) > h\}$
$\mathbf{B} \leftarrow \text{Blob Detection}(\mathbf{M}_{PupilSeg})$
$\mathbf{b} \leftarrow \text{argmax}_{\tilde{b} \in \mathbf{B}} \text{HullArea}(\tilde{b})$
if $\text{HullArea}(\mathbf{b}) < t_b \cdot \sum_{\tilde{b} \in \mathbf{B}} \text{HullArea}(\tilde{b})$ **then** ▷ Merge blobs
    $\mathbf{b} \leftarrow \mathbf{b} \cup \{\tilde{b} \in \mathbf{B} \mid \|\text{Centroid}(\mathbf{b}) - \text{Centroid}(\tilde{b})\| < d\}$
end if
$\mathbf{C} \leftarrow \text{Convex Hull Contour}(\mathbf{b})$
$\mathbf{C} \leftarrow \text{Remove Close Points}(\mathbf{C})$
$\mathbf{C} \leftarrow \text{Remove Colinear Points}(\mathbf{C})$
$\{p, e_x, e_y, \phi\} \leftarrow \text{Ellipse Fit}(\mathbf{C})$
return $\{p, e_x, e_y, \phi\}$

Algorithm 5 Detect occluded pupil $(\mathbf{I}, \mathbf{M}_L)$

$\mathbf{I} \leftarrow 2 \cdot \mathbf{I} - G * \mathbf{I}$
$\mathbf{I} \leftarrow F_{\text{Min}}(\mathbf{I}_{Filt}, k_{Min})$
$\bar{\mathbf{I}} \leftarrow \text{Normalize}(\mathbf{I})$
$\mathbf{M}_{PupilSeg} \leftarrow \{p \in \Omega \mid \bar{\mathbf{I}}(p) > t_{Pupil}\}$
$\mathbf{B} \leftarrow \text{Blob Detection}(\mathbf{M}_{PupilSeg})$
$\mathbf{C} \leftarrow \text{Convex Hull Contour}(\mathbf{B})$
$\mathbf{C} \leftarrow F_{\text{Erode}}(\mathbf{C})$
$\{p, e_x, e_y, \phi\} \leftarrow \text{Ellipse Fit}(\mathbf{C})$
return $\{p, e_x, e_y, \phi\}$

$\mathbf{H}$ on the inverted input image $\mathbf{I}_{BP} = 1 - \mathbf{I}$ with 64 bins. A median filter of size $k_{HistMedian} = 2$ removes outliers. Marking pixels brighter than a threshold h separates the pupil well. Following our observations, we set h to be the grayvalue belonging to the brightest local minimum within $\mathbf{H}$ (Fig. 8b, red bar in histogram).

Next, we want to clean up the derived pupil pixels and perform a blob detection ($\mathbf{B}$) to find connected components. Inspired by Chen *et al.* [15], we work on the convex hull for every blob in $\mathbf{B}$ to remove residues of the glint removal. In difference to [15], we check if the blob detection already detected the pupil. We assume this to be the case if the largest convex hull of each blob covers more than 70 % of the summed area of all blobs. Otherwise, we merge blobs whose center is closer to the center of the largest blob (Fig. 8c) than half the maximum extent of the largest blob d. The contour of the convex hull of the merged blobs then gives us a first estimate of the pupil contour $\mathbf{C}$ (Fig. 8d). We refine this contour by first removing any point closer than 5 pixels to each other and secondly removing colinear points since those are probably generated by the (mostly) straight geometry of the eye lid (Fig. 8e). Finally, we fit an ellipse to the remaining contour points to obtain position p, eccentricity e_x and e_y and angle ϕ of the projected pupil (Fig. 8f).

Partially Occluded Pupil The last case to treat is a strongly occluded pupil (Alg. 5). Here, we boost the contrast using unsharp masking; $\mathbf{I} = 2 \cdot \mathbf{I} - G * \mathbf{I}$ (Fig. 8h), where G is a Gaussian and $*$ the convolution operator. We then apply a minimum filter with radius $k_{Min} = 21$ pixels to remove eye lashes. By normalizing the input image $\mathbf{I}$ to the range $[0,1]$, we can detect pupil segments by an adaptive thresholding. Setting the threshold to $t_{Pupil} = 0.12 + (\|\tilde{p} - p_E\|)^{0.5}$ derives an approximate mask of the pupil fragments, where p_E is the pixel position of the center of the eye ball. We chose this formula to countervail an observed vignette effect at the border of the eye.

As in Alg. 4, we perform the blob detection and merge the resulting blobs to estimate the convex hull of the result (Fig. 8h). To counteract the minimum filter, we erode the result with a similar kernel of size k_{Min}. Finally, we extract the contour of the blob and again perform an ellipse fitting to obtain the ellipse parameters of the projected pupil (Fig. 8j).

6. APPLICATIONS

We implemented several application for our HMD with integrated eye tracker, based on the freely-available Unreal Engine and game content [4]. **Adaptive Depth of Field Rendering** We simulate the accommodation reflex inspired by previous studies for desktop applications [25, 32]. In reality, accommodation allows us to focus on objects at arbitrary distances by flexing our eye lens. In consequence, other objects are naturally blurred. To compute the focus distance, we cast a ray into the scene starting at the viewpoint and directed by the estimated viewing direction from our gaze estimation of one eye. We then determine the distance to the surface the ray hits first (Fig. 9) and render the scene with the appropriate depth-of-field effect turned on. When thin objects are very close to the viewer a binocular eye tracking is required, where the convergence point of both viewing rays defines the focal distance. Nonetheless, in most scenarios the increase of computational effort seems unnecessary.

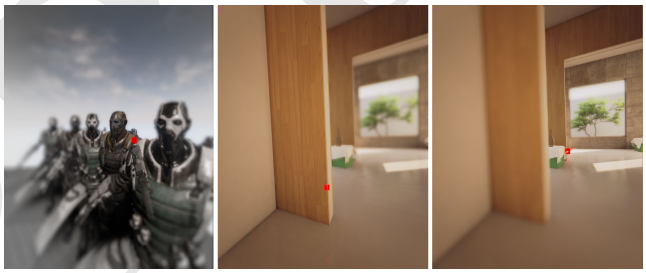

Figure 9: **Real-time Gaze-contingent Rendering.** Foveated rendering (left): Rendering quality and saturation is decreased for peripheral vision. Adaptive depth-of-field effect (center, right): near and far focus distances. Gaze vector shown as red marker.

Foveated Rendering In the second application, we show that our gaze tracker enables simulation of a gaze-contingent display. Previous work showed the potential of this techniques [35, 19]. Due to the rapid acuity fall-off from foveal to peripheral vision rendering could massively benefit from Gaze-contingency. We demonstrate the effect by rendering the scene with five different resolutions describing circles of different radii on the screen. The highest resolution is used in the foveal region where the user is looking at. The render resolution is reduced by a factor of two for each following circle. We smoothly blend between each render resolution to avoid visible resolution seams.

Our current implementation is just a simulated Foveated Rendering and does not lead to an actual performance boost for the renderer we employed. When using a path tracer the number of samples per pixel could be reduced in the peripheral vision whereas in a rasterizer a lower level-of-detail or less texture lookups could be performed (Fig. 9).

Additionally, similar techniques could be used to simulate various visual field defects, such as hemianopia (partial blindness), color blindness, retinitis pigmentosa (night blindness, blurring of vision, loss of central vision, and others) or pigmentary retinopathy (deposits of pigments) (Fig. 9, left).

DRAFT

Gaze Transfer for Avatars In this application, we enhance immersion by mapping the gaze direction and head movements of the user onto an avatar standing virtually in front of him. The eyes of the avatar rotate as the user rotates his eyes and the avatar blinks as the user blinks (Fig. 10). This increases perceived realism for every entity in VR and offers novel opportunities for self-expression. Gaze transfer can be a valuable extension for telepresence applications or user-to-user communication within VR applications.

Gaze Maps are an effective visualization of the user's gaze over time and an effective tool for user experience studies [36]. For a demonstration using our binocular eye tracker, we implemented a player for stereoscopic movies and recorded the gaze for multiple users when watching the video. The gaze maps have been derived by plotting and filtering the estimated screen positions for all the viewers. The result is shown for one frame of the movie in Fig. 11. Most viewers fixated the person in foreground as well as the picture in the background and the table. We used a temperature color coding for visulization (hot areas are fixated more than cool areas).

Figure 10: **Gaze Transfer and Avatar Animation.** (Left, center) Eye tracking enables more expressive and natural character animation. The estimated pupil size and blink event can also be used to animate eye adaptation and blinks instantly (right).

Figure 11: **Gaze visualization.** Gaze maps for images or videos show the fixated display area averaged over time or users. Temperature color scheme represents fixation quantity.

7. EVALUATION

In this section, we evaluate our method by estimating tracking quality and performance. We tested our pupil detection algorithms against two other state-of-the-art algorithms [31, 15]. We conclude the section with a user study with 33 participants.

Performance Evaluation We have implemented our eye tracking framework in C++ using the OpenCV algorithm library [7]. Our primarily CPU-based processing pipeline achieves a total end-to-end latency from capturing the eyes by the cameras until a rendered frame is visible to the user of 32 ms on current hardware (i7-4930K @ 3.4 Ghz, GeForce GTX 780 Ti). The pupil estimation of both eyes and the camera capture threads run in parallel on multiple cores of the CPU. Some of the preprocessing filters

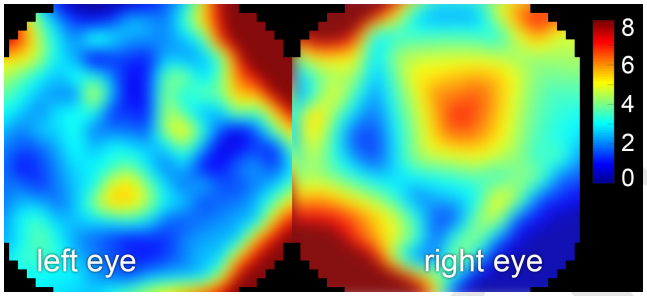

Figure 12: **Gaze direction error.** The absolute error for both eyes over the available FOV, given in screen pixels.

(sharpening, blur) run in CUDA on GPU. The eye-tracking camera resolution is 640×480 pixels at 75 frames per second. Timings for each step of the processing pipeline are given in Table 1.

Process step	*Duration (milliseconds)*
Frame grabbing (@75 Hz)	≈ 13
Pupil estimation	≈ 9
Gaze estimation	< 1
Rendering (Application)	≈ 10
Total Latency	≈ 32

Table 1: **End-to-end latency estimation.**

Tracking Quality In this section, we evaluate our pupil-tracking algorithm in terms of tracking stability and tracking precision. After having calibrated the eye tracker for the two different male users with corrected to normal vision, we measured the tracking precision. We conducted a fair-minded comparison to the *STARBURST* eye tracking algorithm of Li *et al.* [31] and the auto-threshold algorithm of Chen *et al.* [15] working for near-field eye tracking without relying on the corneal reflection. As input we used the glint-free images as these are required by all algorithms.

T1 Pupil Position and Size We test the pupil position and size objectively against ground-truth data derived from manually created pupil masks of a 1987 frames video recorded with both eye tracking cameras. The error values for pupil position and size are computed by the differences of the extracted pupil-ellipse position and eccentricity. The result of the test is summarized in Table 2. The pupil position error ε_{Pos} is computed as the average pixel deviation of the computed position p_e from the reference position p_{gt}:

$$\varepsilon_{Pos} = \frac{1}{n}\sum_{i=1}^{n}\left(\left|p_e - p_{gt}\right|\right) \tag{1}$$

and the pupil size error by the equation

$$\varepsilon_{Size} = \frac{1}{n}\sum_{i=1}^{n}\left(\left|e_x - e_{x,gt}\right| + \left|e_y - e_{y,gt}\right|\right), \tag{2}$$

where e_x, e_y are the eccentricities of the estimated ellipse. In terms of accuracy, our algorithms clearly outperforms the competitors as they can hardly deal with partially occluded pupils in which case our algorithm clearly stands out. Independent from the used pupil detection algorithm the pupil size is closest to the real pupil size for a central view. The pupil size artificially increases as the view tilts towards the sides due to the increasing lens distortion resulting in partial magnification of the projected pupil.

PREPRINT

Test	ε_{Size} (px)	ε_{Pos} (px)
Ours	**0.04**	**2.16**
Auto-threshold [15]	0.63	21.67
Starburst [31]	0.24	13.15

Table 2: **Pupil position and pupil size accuracy.**

T2 Gaze Direction Error We also evaluate the difference of the screen position returned by the eye tracker and the reference screen position set by a visible marker on screen as

$$\varepsilon_{Screen} = \frac{1}{n}\sum_{i=1}^{n}\left|s_e - s_{gt}\right| \qquad \varepsilon_{Ang} = \tan^{-1}\frac{\varepsilon_{Screen}}{d_{EyeScreen}} \tag{3}$$

where s_e and s_{gt} are the estimated and reference screen positions and n the number of tracking samples (n = 30 in our test). The pixel error is then transformed into the angular error by estimating $d_{EyeScreen}$ via ray tracing using the calibrated model. The error is evaluated for thirty different positions.

The error ranges from $\varepsilon_{Ang} \approx 0.5\,^{\circ}$to $\varepsilon_{Ang} \approx 3.5\,^{\circ}$, being generally higher at the borders of the screen due to stronger occlusion and therefore aggravated pupil position estimation. The interpolated screen position error is visualized in Fig. 12.

User Study We tested our eye tracker with 33 participants (25 males, 8 females); 15 had normal vision, but 18 had corrected-to-normal vision. The current prototype does not support wearing glasses when using the HMD. However, the lenses can be adjusted to compensate for a wide variety of ametropia and hypertropia [17]. Every person started with the user calibration procedure and then was able to use the Adaptive Depth-of-Field application (Sec. 6).

Afterwards, we asked the users to rate certain aspects of the device (update rate, latency, stability, accuracy) and the application (naturalness, usefulness, user experience). The complete questionnaire and evaluated numbers are included in the supplemental material. The evaluation of the user feedback is visualized in Fig. 13.

Summarizing the results the user feedback was very positive with regard to the user experience and evaluation of the usefulness of the system. Every user mentioned that they can think about and want gaze usage in many application using the presented HMD.

The stability and accuracy was rated positive but not completely convincing yet. There were two major issues, which explain the reduced rating. The system is currently an early prototype, and it includes disturbing redundant cables from the cameras, as well as an inflexible display cable, which resulted in slight shifts of the HMD when turning the head, which reduced the accuracy of the gaze estimation. Another issue for some participants was the usage of mascara on the eye lashes which negatively influenced our pupil estimation resulting in a reduced user experience.

8. DISCUSSION AND FUTURE WORK

Limitations The concept of adjustable lens controllers provides a sharp vision even for people usually wearing glasses. Wearing glasses inside the HMD is an open problem as this would require larger lens-to-eye distance, larger lenses, and a larger screen for the same view. A strict positioning of the HMD with respect to the head is also crucial. Otherwise a recalibration becomes necessary.

Long-term user study In this work, we tested the tracking quality of our gaze estimation algorithm just for a small number of people and a limited time (several minutes) of usage. In the future, we plan a larger user study to improve the hardware design and software of our prototype. Additional studies with longer usage sessions will provide more information about robustness, usability and wearing comfort.

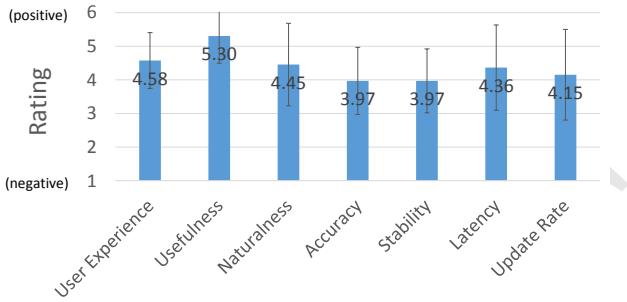

Figure 13: **User study results.** Blue bars show user ratings concerning specific aspects (entitled on the left). Scale ranging from [1, 6] (negative/positive). Black bars represent standard deviation.

Auto-calibration In the literature there are software methods for auto-calibration that rely on natural scan pathes of the environment and provide a seamless transition between calibration and interaction phase [34]. These concepts may be beneficial for our system. However, to the best of our knowledge none of those methods have been tested within an HMD yet. Alternatively, by using additional hardware, it might be possible that the calibration process can be largely simplified or completely automated. Klefenz et al. and Kohlbecher et al. exploit the precalibration in a stereoscopic camera setup to track the pupil without additional user calibration [28, 29]. Alternatively a depth sensor could provide valuable information about the actual anatomy of the individual eye. An automated calibration process seems interesting, even if additional hardware would increase weight and complexity of the device.

Applications With the presented applications we only scratch the surface of possible VR scenarios. Many other applications are enabled with instant knowledge of the users gaze or at least could benefit from this input, e.g., gaze-based selection and manipulation or studies on user interfaces.

Perception studies can assess the potential of eye tracking to exploit or examine the perception of the human visual system or an individual user. These insights may lead to methods, which will improve viewing experience or accelerate the rendering process. Perception studies also enable evaluation of simulations in VR, e.g., in the fields of assembly processes or training for aerospace, military or surgery, psychological therapy, or eye disease simulation.

Using eye tracking as an input device enables novel gaze-based interaction metaphors, hands-free interaction with Attentive User Interfaces (AUIs) or an additional communication channel. The user is able to express his interest naturally by gaze. With additional cameras our HMD prototype could be extended for Augmented Reality usage (AR) where hands-free interaction is beneficial and precise IPD estimation and calibration are very important. Instead of using a closed body, the mirror-based setup, and gaze-estimation technique could also be used for See-Through HMDs.

9. CONCLUSION

We have presented a complete binocular eye-tracking solution for head-mounted displays. Our system relies on low-cost components that should be affordable for every user group. This aspect opens the door for a large variety of novel applications and contributes to progress in research. The prototype has been tested by a small group of subjects. For the future, we will work with a larger group of people in order to improve pupil detection and user comfort. We also plan to investigate new ways for continuous and automatic user calibration.

PREPRINT

10. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 256941, Reality CG. We thank Lorenz Rogge, Pablo Bauszat and Günther Stengel for supporting us with the project. We also thank Epic Games, Inc. for making the Unreal Engine and the stock footage openly available.

11. REFERENCES

[1] Agisoft PhotoScan, 2014. agisoft.com , vis. 12-20-2014.

[2] metaio SDK, 2014. dev.metaio.com/sdk, vis. 1-5-2015.

[3] Arduino microprocessor, 2015. arduino.cc, vis. 03-12-2015.

[4] Epic Games, Inc., Unreal Engine, 2015. unrealengine.com, vis. 03-12-2015.

[5] Next Limit S.L., Maxwell Render, 2015. maxwellrender.com, vis. 03-12-2015.

[6] Oculus VR Oculus Rift, 2015. oculus.com, vis. 03-12-2015.

[7] OpenCV Library, 2015. opencv.org, vis. 03-12-2015.

[8] M. Abrash. What VR could, should, and almost certainly will be within two years. Steam Dev Days, Seattle, 2014.

[9] F. H. Adler, P. L. Kaufman, L. A. Levin, and A. Alm. *Adler's Physiology of the Eye*. Elsevier Health Sciences, 2011.

[10] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Machine Learning in Computer Vision: What, Why and How?*, 1993.

[11] P. Bazanov and T. Järvenpää. Gaze estimation for near-eye display based on fusion of starburst algorithm and fern natural features. In *FRUCT 2011*, pages 1–8, 2011.

[12] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *CVPR 2001. Proceedings*, volume 1, pages I–355. IEEE, 2001.

[13] J.-Y. Bouguet. Camera calibration toolbox for matlab, 2010. vision.caltech.edu/bouguetj, vis. 08-10-2014.

[14] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev. *3DUI: theory and practice*. Addison-Wesley, 2004.

[15] S. Chen and J. Epps. Efficient and robust pupil size and blink estimation from near-field video sequences for human machine interaction. *IEEE Transactions on Cybernetics*, 44(12):2356–2367, Dec 2014.

[16] T. Dera, G. Boning, S. Bardins, and E. Schneider. Low-latency video tracking of horizontal, vertical, and torsional eye movements as a basis for 3dof realtime motion control of a head-mounted camera. In *SMC'06, Proceedings*, volume 6, pages 5191–5196. IEEE, 2006.

[17] J. E. Doble, D. L. Feinberg, M. S. Rosner, and A. J. Rosner. Identification of binocular vision dysfunction in traumatic brain injury patients and effects of individualized prismatic spectacle lenses. *PM&R*, 2(4):244–253, 2010.

[18] A. Duchowski. *Eye tracking methodology: Theory and practice*, volume 373. Springer, 2007.

[19] A. T. Duchowski and A. Çöltekin. Foveated gaze-contingent displays for peripheral lod management, 3d visualization, and stereo imaging. *TOMCCAP'07*, 3(4):6, 2007.

[20] A. T. Duchowski, V. Shivashankaraiah, T. Rawls, A. K. Gramopadhye, B. J. Melloy, and B. Kanki. Binocular eye tracking in virtual reality for inspection training. In *ETRA'00, Proceedings*, pages 89–96. ACM, 2000.

[21] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[22] A. W. Fitzgibbon, R. B. Fisher, et al. A buyer's guide to conic fitting. *DAI*, 1996.

[23] D. W. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *PAMI'10*, 32(3):478–500, 2010.

[24] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[25] S. Hillaire, A. Lécuyer, R. Cozot, and G. Casiez. Using an eye-tracking system to improve camera motions and depth-of-field blur effects in virtual environments. In *VR'08, Proceedings*, pages 47–50. IEEE, 2008.

[26] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press, 2011.

[27] K. Holmqvist, M. Nyström, and F. Mulvey. Eye tracker data quality: what it is and how to measure it. In *ETRA'12, Proceedings*, pages 45–52. ACM, 2012.

[28] F. Klefenz, P. Husar, D. Krenzer, and A. Hess. Real-time calibration-free autonomous eye tracker. In *ICASSP'10, Proceedings*, pages 762–765. IEEE, 2010.

[29] S. Kohlbecher, S. Bardinst, K. Bartl, E. Schneider, T. Poitschke, and M. Ablassmeier. Calibration-free eye tracking by reconstruction of the pupil ellipse in 3d space. In *ETRA'08, Proceedings*, pages 135–138. ACM, 2008.

[30] N. Kumar, S. Kohlbecher, and E. Schneider. A novel approach to video-based pupil tracking. In *SMC'09, Proceedings*, pages 1255–1262. IEEE, 2009.

[31] D. Li, D. Winfield, and D. J. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *CVPR 2005, Proceedings of*, pages 79–79. IEEE, 2005.

[32] M. Mauderer, S. Conte, M. A. Nacenta, and D. Vishwanath. Depth perception with gaze-contingent depth of field. In *CHI'14, Proceedings*, pages 217–226. ACM, 2014.

[33] F. Moisy. Ezyfit: a free curve fitting toolbox for matlab. *U. Paris Sud. Version 2*, 2011.

[34] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *UIST'13, Proceedings*, pages 261–270. ACM, 2013.

[35] E. M. Reingold, L. C. Loschky, G. W. McConkie, and D. M. Stampe. Gaze-contingent multiresolutional displays: An integrative review. *HFES Journal*, 45(2):307–328, 2003.

[36] A. Schall. *Eye tracking in user experience design*. Morgan Kaufmann, 2014.

[37] E. Schneider, T. Villgrattner, J. Vockeroth, K. Bartl, S. Kohlbecher, S. Bardins, H. Ulbrich, and T. Brandt. Eyeseecam: An eye movement–driven head camera for the examination of natural visual exploration. *Annals of the NY Academy of Science*, 1164(1):461–467, 2009.

[38] K.-H. Tan, D. Kriegman, and N. Ahuja. Appearance-based eye gaze estimation. In *WACV 2002, Proceedings*, pages 191–195, 2002.

[39] C. Topala and C. Akinlara. An adaptive algorithm for precise pupil boundary detection using the entropy of contour gradients. 2013. Elsevier preprint.

[40] O. Williams, A. Blake, and R. Cipolla. Sparse and semi-supervised visual mapping with the S3GP. In *CVPR'06, Proceedings*, volume 1, pages 230–237, 2006.

PREPRINT