Computers & Graphics

Published paper version:
https://doi.org/10.1016/j.cag.2020.06.007

SalientGaze: Saliency-based gaze correction in virtual reality

Peiteng Shi[a,b,*], Markus Billeter[a], Elmar Eisemann[a]

[a] *Computer Graphics and Visualization Group, Delft University of Technology, Delft, 2628 XE, the Netherlands*
[b] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China*

ARTICLE INFO

Article history:
Received 25 March 2020
Accepted 30 June 2020
Available online 9 July 2020

Keywords: Virtual Reality, Eye-tracking, Headsets Shifts, Saliency, Stereo, Drift Estimation

ABSTRACT

Eye-tracking with gaze estimation is a key element in many applications, ranging from foveated rendering and user interaction to behavioural analysis and usage metrics. For virtual reality, eye-tracking typically relies on near-eye cameras that are mounted in the VR headset. Such methods usually involve an initial calibration to create a mapping from eye features to a gaze position. However, the accuracy based on the initial calibration degrades when the position of the headset relative to the users' head changes; this is especially noticeable when users readjust the headset for comfort or even completely remove it for a short while. We show that a correction of such shifts can be achieved via 2D drift vectors in eye space. Our method estimates these drifts by extracting salient cues from the shown virtual environment to determine potential gaze directions. Our solution can compensate for HMD shifts, even those arising from taking off the headset, which enables us to eliminate reinitialization steps.

1. Introduction

Virtual Reality (VR) is an exciting technology that has reached the consumer market with affordable head-mounted displays (HMDs). Nowadays, VR is successfully applied in many different tasks but some challenges remain. Real-time rendering of high-quality content is an example, as very high framerates are needed and VR display resolutions increase constantly. Further, new paradigms for user interaction are required. Traditional input devices like a mouse are difficult to employ and any physical peripherals are obscured by the HMD.

Eye-tracking with gaze estimation provides a compelling direction to address rendering bottlenecks and interaction. Foveated and perceptually-driven rendering focuses resources on regions currently observed by users and gaze information can be used to interact with virtual environments [1, 2, 3]. The latter is valuable for a person with physical disabilities [4]. Many applications can benefit from gaze estimation, e.g., realistic eye movements for avatars [5], interactive tone mapping [6], or just to acquire knowledge about what users are observing or actively focusing on [7].

However, there are still problems related to eye-tracking in VR. Firstly, the gaze estimation in a head-mounted display is very sensitive to HMD movements relative to the users' head [8, 9]. Such movements happen regularly, but are especially noticeable when a user touches the HMD to adjust its position or even temporarily removes it completely. Movements of the HMD can result in large errors in the gaze estimation unless these shifts are accounted for by the gaze estimation method. Further, many eye-tracking methods require a calibration process [10, 11, 12] that users need to perform before they can start with a VR experience or even, occasionally, during execution.

In this paper, we show that inaccurate gaze estimation due to shifts and movements can be sufficiently corrected with a drift vector that is applied to the detected pupil locations. We present a method to detect this drift vector when using near-eye cameras for tracking by utilizing saliency information of the displayed 3D scene. It is more likely that users focus on salient elements, and we can use this assumption to detect drifts in

*Corresponding author:
e-mail: p.shi@tudelft.nl (Peiteng Shi)

the calibration on the fly, without interrupting the use of VR HMDs. Contrary to other solutions, our method does not involve any prior knowledge about users nor rely on large amounts of training data.

We test our method in multiple 3D scenes where users perform distinct tasks. Specifically, our main contributions are:

- We show that, in an interpolation-based gaze estimation method, HMD shifts can be compensated for with 2D translations applied to the pupil location.
- We present an auto-calibration method deriving this 2D correction vector based on saliency detection applied to the virtual content.

Section 2 introduces the topic and related work. We then present our method in Section 3. Section 4 describes the experimental results from a user study. Section 5 concludes the paper and discusses future research.

2. Related work

Given their wide applicability, eye-tracking and gaze estimation have been explored thoroughly, and we refer readers to the survey of Kar and Corcoran [13], as well as the summary of Hansen and Ji [14] for details.

Video-based eye-tracking systems, as used in this paper, operate in the near infrared spectrum. These systems illuminate the eyes using (invisible) infrared LEDs and capture images with an infrared camera to estimate the gaze. The concurrently-developed machine learning approach presented by Kim et al. [9] operates directly on these images, using them to infer gaze estimates via a neural network. However, like other machine learning-based methods [15, 16], they need a large amount of training data. They propose to synthesize the data, but this nevertheless remains a costly endeavor, taking significant computational resources. Other learning-based approaches avoid eye-tracking cameras, and instead rely on other inputs. For example, Soccini [17] inputs head movements and image features into a convolutional neural network that estimates gaze positions. Xu et al. [18] simplify this further for 360° videos by only using previously estimated gaze positions together with image content.

Many other gaze estimation methods rely on specific features, like pupil positions or *glints* (reflections on the eye cornea surface) [19]. Stengel et al. [20] use a virtual model of the optical system comprising the eyes and compute light paths through this optical system to map pupil positions to gaze. In this method, a precise optical model of the VR HMDs has to be constructed, making the method device-dependent.

PupilLabs [21] is a commercial example of a calibration-based system. Here, users are asked to focus on a number of calibration markers distributed over a large portion of the user's field of view. The calibration yields a number of pupil- and screen-position correspondences, from which a mapping between screen- and gaze-space is derived. This paper focuses on such interpolation-based gaze estimation approaches. These have the advantage that no prior knowledge of HMDs or eye-tracking systems is required but the disadvantage that the method becomes inaccurate as the relative position between the HMD and the user's head changes. While it is possible to just re-run the calibration process, this would be invasive, negatively affecting the users' immersion. Our solution avoids this issue.

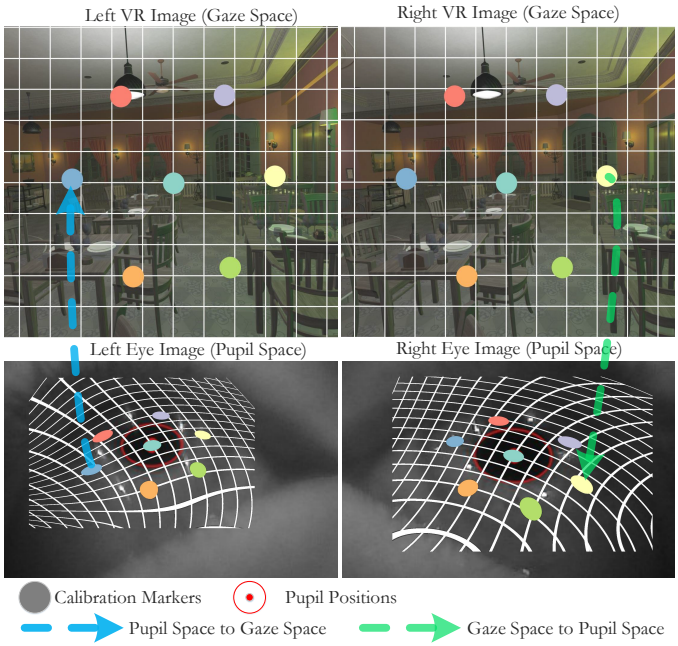


Fig. 1: Interpolation-based gaze estimation methods create a mapping between pupil positions (in pupil space) and gaze positions (in gaze or screen space) by measuring pupil positions corresponding to displayed calibration markers (gaze space). Points inside of the region spanned by the calibration markers are found through interpolation. The grid and the colored calibration markers show how each position in gaze- and pupil space map to each other using a polynomial function for interpolation.

Our method relies on saliency information [22, 23] of the displayed images to identify HMD shifts. Sugano et al. [10, 11] first applied saliency-based analysis in a desktop eye-tracking system. Their approach collects eye images and corresponding saliency maps, and uses these as input to train a mapping to a gaze position. They also combine this solution with electrooculography (EOG) data in a head-mounted setting [12]. While self-calibrating, the main drawback is the long data collection time that is required.

Chen and Ji [24] combine information from saliency images with a 3D eye model to perform auto-calibration, achieving their goal of avoiding a time-consuming initial setup in desktop eye-tracking systems. They treat saliency images as probabilistic information, and estimate free variables in their eye model via an optimization step. Perra et al. [25] consider the case, where an initial calibration of a head-worn device becomes invalid over time. Their approach subdivides the saliency images into several regions and considers the salient region closest to the current gaze estimation as the regarded position. Tripathi and Guenter [26] instead rely on matching eye trajectories to the motion of rendered objects to estimate the gaze position. Sidenmark [27] finds matching points based on user interactions, but explores this mainly as an alternate method for acquiring data for the initial calibration.

Our gaze estimation builds upon a standard interpolation-

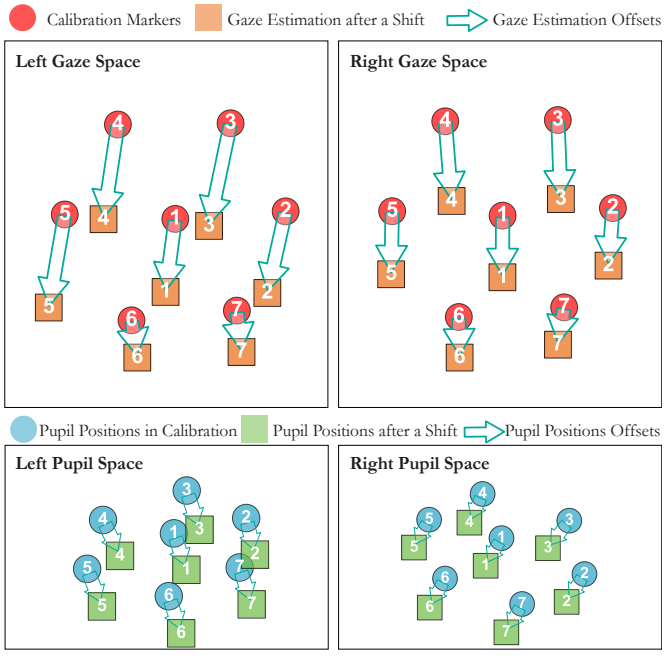


Fig. 2: Gaze and pupil positions before and after a shift. The red disks denote the positions of the calibration markers in gaze space. Blue disks show the corresponding pupil positions in pupil space. After a shift of the HMD, the same calibration markers give us a new set of pupil positions, shown as green squares. Using the old, uncorrected gaze estimation with the green squares as input would result in the orange squares. The offsets between the sets of positions in the gaze space vary both in direction and magnitude. However, the offsets in pupil space are very coherent, having both similar direction and magnitude, and motivate our 2D drift vector model in pupil space.

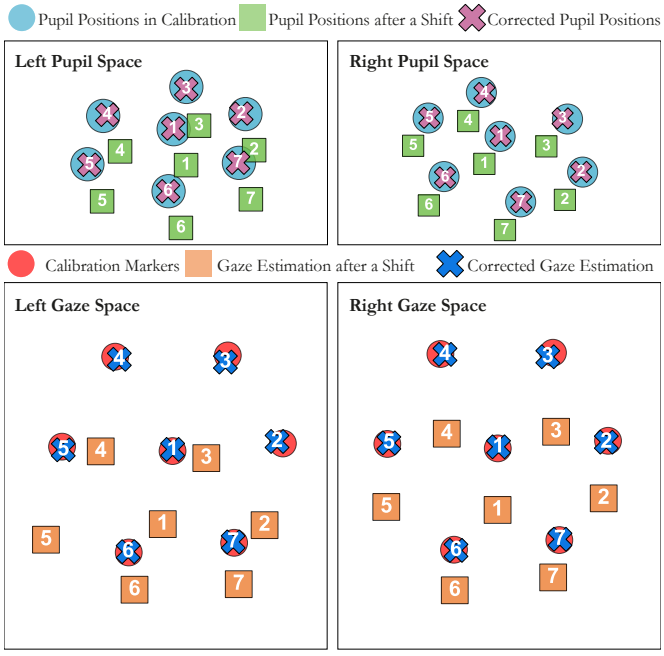


Fig. 3: Correction using the error-minimizing 2D drift vector in pupil space. Similar to Figure 2, the red disks show the original calibration markers and blue disks show the corresponding pupil positions. After a shift, the green squares show the new pupil positions corresponding to the calibration markers, and the orange squares show the gaze positions using the uncorrected estimation. An error-minimizing 2D drift vector would map the new pupil positions (green squares) close to the original locations (purple crosses in pupil space), and using these as an input to the original gaze mapping function results in the estimates shown as blue crosses in the gaze space.

based method [21]. The selected method performs an initial calibration using seven markers, as shown in Figure 1, which cover roughly 90% of the users' visible field of view when looking at the center of the screen in VR. A mapping of the form $\mathbf{g} = G(\mathbf{p})$ is fitted to these measured points, where $\mathbf{g}$ is the estimated gaze point, $\mathbf{p}$ is the 2D pupil position in the view of the eye-tracking camera, and $G = (G_x, G_y)$ with G_x and G_y each being fourth degree polynomials in x and y of form $a_1 + a_2\,x + a_3\,y + a_4\,x^2 + a_5\,y^2 + a_6\,x\,y + a_7\,x^2\,y^2$ (i.e., with seven free parameters each). After calibration, G maps 2D pupil positions to image positions on the VR screens (each eye has a separate G). Inverting G gives the reverse mapping. The calibration process ensures that G is locally one-to-one and thus invertible. Contrary to previous approaches, where G would be static, our proposed method adapts G to correct for drifts during HMD usage.

3. The SalientGaze method

Our method, *SalientGaze*, consists of two parts. We first discuss our HMD drift model and introduce 2D drift vectors, which we show to be sufficient for correcting the initial calibration over time. We then present a saliency-based method to compute the correction vector on-the-fly, during HMD use.

3.1. 2D drift vector model

An HMD can shift due to head movements or due to the user touching the device, e.g., when adjusting for comfort or even when taking off the device for a moment and reseating it afterwards. Once a change in relative position occurs, the initial gaze mapping function G is no longer accurate. Figure 2 (top row) illustrates an example. One can observe that the resulting errors in the gaze estimation vary in direction and magnitude over the entire domain. However, when analyzing the shift in the near-eye camera view, referred to as *pupil space* in this paper, we observe that the offsets between the corresponding pupil positions are very similar in direction and magnitude (Figure 2, bottom row). Therefore, we propose to correct a shift with a single 2D translation in pupil space.

Formally, the 2D translation, the *drift vector* $\mathbf{d}$, is applied as follows:

$$\mathbf{g_d} = G(\mathbf{p} - \mathbf{d}), \tag{1}$$

where $\mathbf{g_d}$ is the corrected gaze position and $\mathbf{d}$ the shift. To evaluate the validity of using a single 2D drift vector for correcting the calibration. For a given vector $\mathbf{d}$, we measure the error as the sum of distances between the individual calibration markers $\hat{\mathbf{g}}$ and the corresponding gaze estimate $\mathbf{g_{\hat{d}}}$:

$$E = \sum_{k=1}^{N} \left\| \mathbf{g_{\hat{d}}} - \hat{\mathbf{g}} \right\| \tag{2}$$

where $N = 7$ is the number of calibration makers (see Figure 3).

We will first illustrate the possible improvement that a correction with a single drift vector can have. To this extent, given an initial calibration (defined by G), we compute an *error-*

minimizing drift vector, $\hat{\mathbf{d}}$, that minimizes Equation 2. In practice, we found that such error-minimizing drift vectors keep the average error below 1.4° on average. The details of the corresponding study and results can be found in Section 4.

3.2. 2D drift vector estimation

Given our finding that a single drift vector $\mathbf{d}$ is sufficient to correct the calibration, it implies that it would be possible to recalibrate on the fly by just determining a single point that a user is looking at. Unfortunately, finding such a point is difficult. We propose an on-the-fly method that relies on the saliency of the displayed VR content to derive the drift vector iteratively.

As we do not control the content of the rendered images, determining the saliency in a single image is not guaranteed to accurately predict the region of attention (e.g., users might just randomly look around). Further, none or even multiple areas might appear salient (see second column of Figure 4 for example saliency images). For these reasons, additional criteria are needed to increase robustness. Our goal is to construct a probability map, from which the estimated drift vector can be derived. This map will be produced by accumulating saliency maps from several well-chosen frames. In the following, we detail the construction of this map.

We filter the input and omit frames with rapid eye-movement, using the I-DT algorithm from Salvucci and Goldberg [30]. For each remaining frame i, the rendered images $I_i^{g,\ell}$ and $I_i^{g,r}$ (in gaze space, identified by the index ^{g}) and associated pupil positions $\mathbf{p}_i^\ell$ and $\mathbf{p}_i^r$ are extracted. Then, saliency maps $S_i^{g,\ell}$ and $S_i^{g,r}$ are computed from the rendered images $I_i^{g,\ell}$ and $I_i^{g,r}$, respectively. We use the boolean map saliency detection method by Zhang and Sclaroff [31], which shows good accuracy in the MIT saliency detection benchmark [32, 33, 34]. For brevity, from here on, the ℓ and r will be omitted whenever operations are applied equally to the left and right eye.

Computing saliency images is expensive and therefore the rendered content is downsampled before saliency detection. Furthermore, our implementation crops the images to a region of interest that roughly matches the center area of the view covered by our (original) calibration markers, as most of the time, users are looking at the center region in VR content [35]. Figure 4 shows the central image regions and the corresponding saliency maps in the two left-most columns.

We interpret the saliency images as instantaneous probabilities, carrying information about the location a user might be looking at. However, the gaze position cannot be reliably estimated from a single frame. Therefore, SalientGaze combines several saliency images over multiple frames. Specifically, we create a map A_i^d, where each pixel represents a drift vector and their values indicate their corresponding likelihood at frame i.

The map A_i^d is constructed through the following steps. We transform the saliency image S_i^g from the gaze/screen space into the pupil space (^{p}), yielding S_i^p, by fetching for each pixel $\mathbf{p}$ in S_i^p the corresponding value from S_i^g. Specifically, $S_i^p(\mathbf{p}) = S_i^g(G(\mathbf{p}))$ (Figure 4, second and third column). As can be seen from the formula, although we hereby map S_i^g into pupil space, we do not actually have to invert the function G to construct S_i^p.

Next, we translate S_i^p such that the current pupil locations are centered (Figure 4, last two columns), resulting in the map S_i^d, whose domain we refer to as drift-vector space. For example, if no drifts have occurred and an observer is looking at a salient object, the salient area of that object would be centered in S_i^d. If drifts have occurred, this salient area would be offset and its location would then indicate the drift vector.

A_i^d accumulates the information over several frames:

$$\begin{aligned} A_i^d &= \sum_{j=0}^{i} \lambda^j S_{i-j}^d \\ &= S_i^d + \lambda A_{i-1}^d. \end{aligned} \tag{3}$$

The parameter $\lambda \in (0, 1]$ controls how much history is kept: a smaller value causes old data to affect the current result less, a higher value will keep historic data longer. Figure 5 shows an example of how A_i^d changes over time.

The location of the maximum in A_i^d gives the current *instantaneous estimate* of $\hat{\mathbf{d}}$ denoted as $\mathbf{d}_i$. We filter the instantaneous estimates to handle high-frequency transients to yield the final estimated drift $\bar{\mathbf{d}}_i$:

$$\bar{\mathbf{d}}_i = \bar{\mathbf{d}}_{i-1} + \frac{\mathbf{d}_i - \bar{\mathbf{d}}_{i-1}}{\left\|\mathbf{d}_i - \bar{\mathbf{d}}_{i-1}\right\|} \gamma\left(\left\|\mathbf{d}_i - \bar{\mathbf{d}}_{i-1}\right\|\right). \tag{4}$$

Figure 6 shows both the instantaneous estimates and the smoothed estimate over time. The main purpose of Equation 4 is to limit the maximum change of our estimate. This is controlled by the function $\gamma(x) = \delta\, e^{-x^2/(2\sigma_\gamma^2)}$. The Gaussian shape acts as a low-pass filter: small changes are unaffected by the filter, whereas changes of larger magnitudes are increasingly suppressed. In particular, δ determines the maximum amount of change in a single step. The parameters λ, σ_γ and δ were determined empirically, and are in practice set to 0.992, 0.2 and 0.003, respectively. We present detailed information on determining the empirical parameters in the supplementary material, which proved robust in our use cases.

3.3. Improving 2D drift vector estimation with stereo

In stereo rendering, images are produced for both the left and the right eye. So far, SalientGaze produces an independent estimate for each eye. The estimation can be improved with the assumption that an observer is generally looking at the same object with both eyes.

It is possible to take advantage of this stereo-consistency in multiple different stages of SalientGaze. Specifically, we explored pre-filtering the saliency images $S_i^{g,\ell}$ and $S_i^{g,r}$ before accumulation as well as filtering the accumulated maps $A_i^{p,\ell}$ and $A_i^{p,r}$ before giving an instantaneous estimate. Based on our experiments, we settled on the latter.

Both choices rely on reprojecting the left eye's map to the right eye and vice-versa [36, 37]. This is possible since depth information of our rendered images is retained. Conceptually, the 3D world position for each pixel in gaze (screen) space of one eye is constructed using the inverse of the eye's camera and projection matrices. Then the world positions are projected to the other eye's gaze space using its transformation matrices.

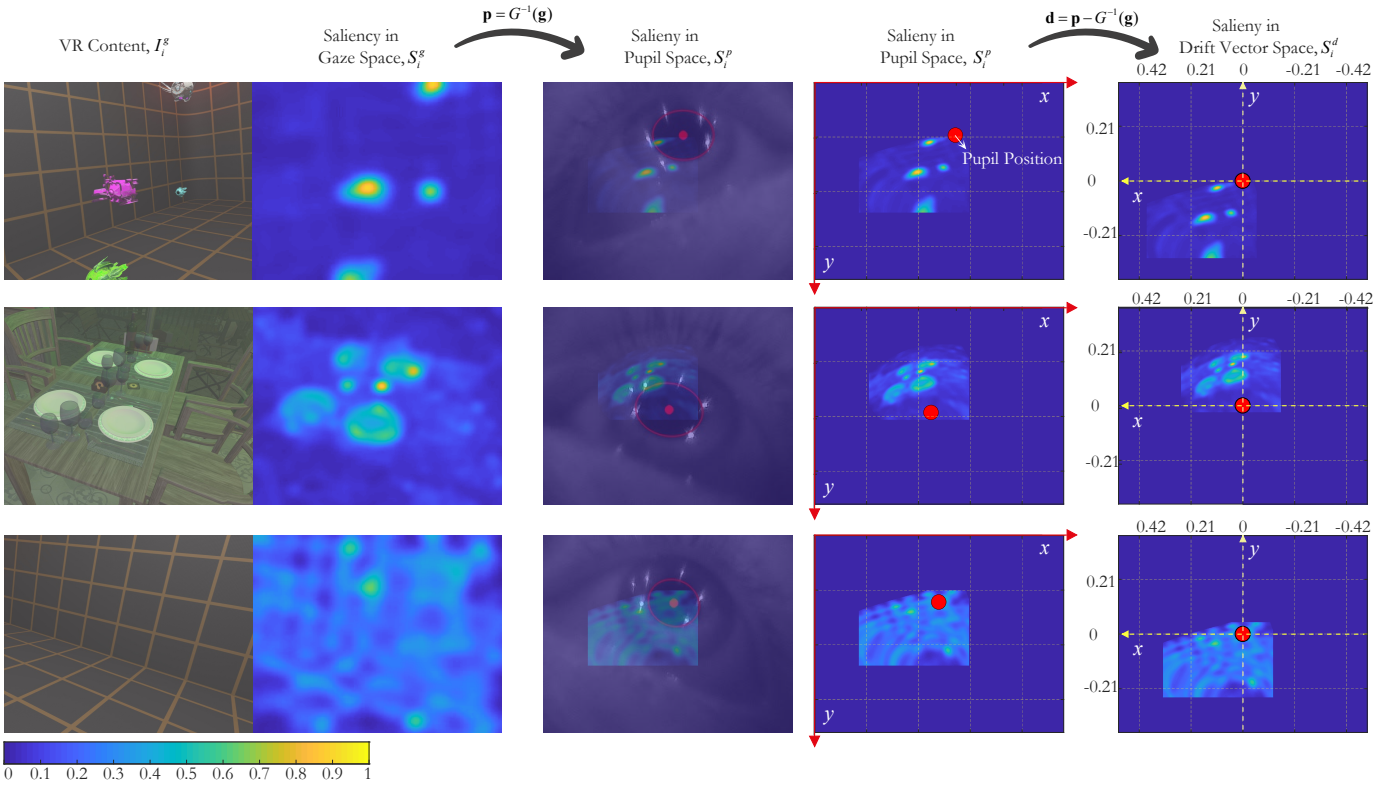


Fig. 4: Saliency map processing pipeline. Processing starts with cutouts from rendered images I_i^g (left column) from filtered frames. Next, the corresponding saliency maps S_i^g (second column from left) are computed. The saliency images are then transformed into pupil space, giving S_i^p. The third column shows S_i^p overlaid onto an image of the eye recorded using the near-eye cameras in the HMD. Finally, the maps are translated such that the detected pupil position (red dots) is centered (rightmost column). This gives us a map S_i^d in drift-vector space, where each pixel carries information about the probability for the corresponding drift vector $\mathbf{d}$. Note that this figure only shows three examples from a single eye - the same process is applied to both eyes independently.

Practically, the transformations can be concatenated into a single matrix for each direction (left to right, right to left). Some reprojected samples will not be visible to the other eye, which implies that the corresponding values should be set to zero. By comparing the depth of the reprojected pixel to the value already in the depth buffer of the target eye, these occlusion cases can be detected.

Once reprojected, we combine the information by multiplying the reprojected map with the original map. For example, for the left eye, we produce a reprojected map $A^{g,\ell\leftarrow r}$ by reprojecting $A^{d,r}$ from the right drift vector space to the left gaze space. We then multiply it pixel-wise with the left eye's map $A^{g,\ell}$ to produce a new combined map $C^{g,\ell}$ in gaze space

$$C_i^{g,\ell} = A_i^{g,\ell\leftarrow r} \odot A_i^{g,\ell}$$

and map it back to drift-vector space (via pupil space), yielding $C^{d,\ell}$. The procedure is repeated for the right eye to produce the corresponding $C_i^{g,r}$. The only change to the previously presented version of SalientGaze is that the instantaneous estimates $\mathbf{d}_i$ are now derived from the $C_i^{d,\ell}$ and $C_i^{d,r}$instead of $A_i^{d,\ell}$ and $A_i^{d,r}$. Figure 7 illustrates the process and shows maps from the various steps.

4. Experimental results and analysis

We illustrate the effectiveness of SalientGaze on an HTC Vive Pro with a binocular eye-tracking system by PupilLabs [21], which consists of two infrared near-eye cameras, capturing images at a rate of 120 Hz and a resolution of 320×240. We use the software supplied with the eye-tracking system to extract the pupil positions. We render 3D scenes at 2036×2260 pixels per eye, as recommended for the HTC Vive Pro (screen resolution is 1440×1600 per eye). The rendering is implemented in OpenGL, and uses an NVIDIA GTX 1080 GPU (with a Intel Core i7 4820K CPU with 24GB of RAM). The rendering maintains ~90Hz in the simpler 3D environments (e.g., used for the game-like task, see Figure 8), and at ~53Hz in the more complex scenes. In the latter, we benefit from SteamVR's motion smoothing mechanism. Our implementation processes VR content at 20Hz, that is, it only considers a subset of the rendered frames.

The saliency detection is applied to a downsampled image roughly one third of the original size (factor of 0.3). Only the central 288×299 pixel region of interest is used. For the accumulation map A_i^d, we use a resolution of 960×720 (3× the eye-tracking device's camera resolution).

4.1. Experimental setup

We design two experiments in the user study:

- In the first experiment, we test SalientGaze's performance with regard to an adjustment of the HMD by the user. We refer to this as the *adjustment experiment*.
- In the second experiment, we let the participants perform the calibration and then save the calibration data. We then

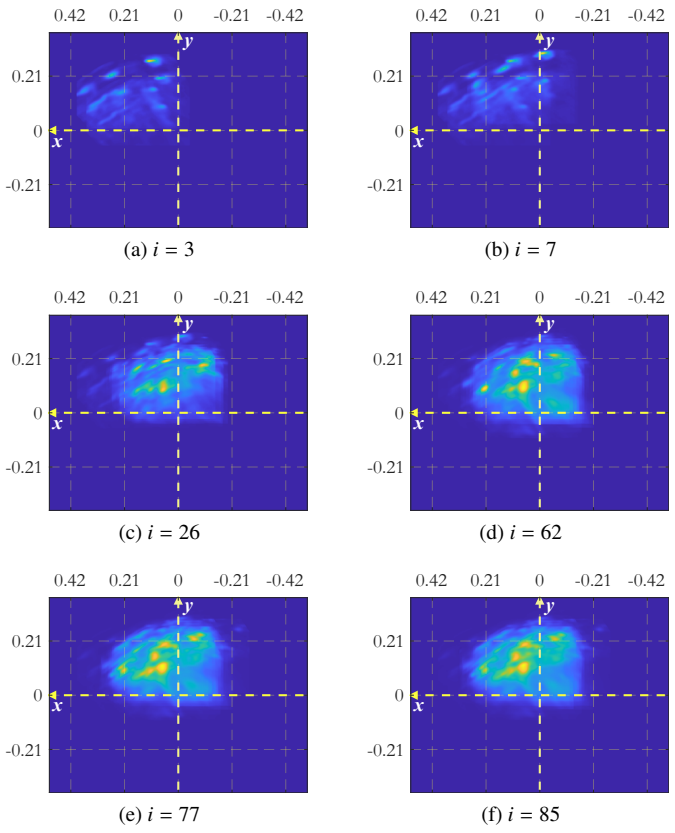


Fig. 5: The accumulation map, A_i^d, shown over time. The index i is the index of the filtered frame (filtered frames occur roughly at a rate of two per second). The maximum position, $\text{argmax}_{\mathbf{d}}\ A_i^d(\mathbf{d})$, indicates the instantaneous estimate of the drift vector $\mathbf{d}$. In this example, the map A_i^d stabilizes over time and results in relatively stable instantaneous estimates.

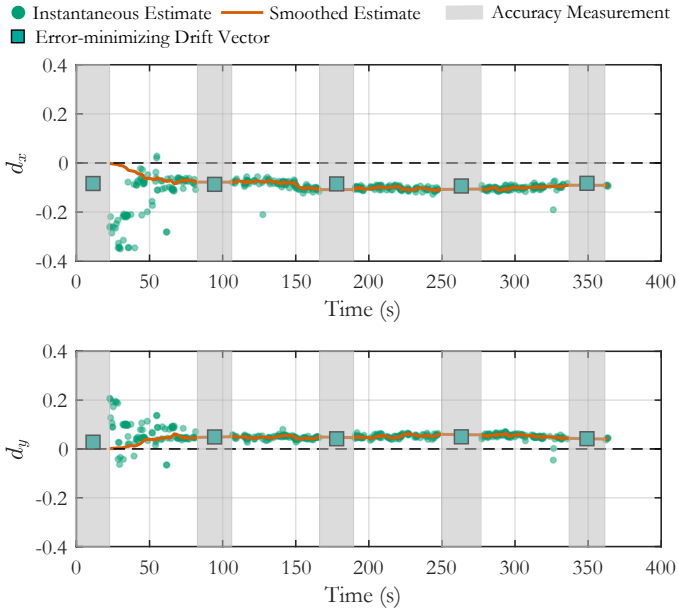


Fig. 6: Example time series of a drift estimation experiment, showing the drift estimates over time. The x- and y-axes of the drifts are shown separately (top and bottom plots, respectively). Green points correspond to the instantaneous estimates based on the accumulation map A_i^d. Early on, the instantaneous estimates are very noisy. These are filtered to produce the smoothed drift estimate $\bar{\mathbf{d}}$, which is shown as a red line. The filtering is clearly observable, and ensures that there is no jump in the returned estimate. The dark gray regions correspond to the periodic accuracy measurements where the ground truth drift is measured (see Section 4 for full details of the experiment setup).

ask the users to remove and subsequently reseat the headset. We refer to this as the *reseating experiment*. Its aim is to test whether initial calibrations can be avoided after a short break or even when beginning another session.

Both experiments are performed in the context of two tasks that model common virtual-reality scenarios:

- The first task consists of a small shooting game, where users have to aim at and shoot small robots/drones in a 3D environment as shown in Figure 8(a). The game consists of several two-minute rounds that are individually scored (scores are continuously visible to users). Users get a positive score for shooting the round robots (Surveillance Drone and K07 Drone) and a negative score for hitting the rectangular Sci-Fi Camera Drone. The number of robots increases with time. While mimicking some popular VR games, the setup is aimed at testing a configuration with few very salient objects that are moving in the scene. This is common in games (e.g., Beat Saber [38] or Space Pirate Trainer [39]) and in some training scenarios.

- The second task consists of two static (but relatively large) 3D scenes, shown in Figure 8(b)), that users are asked to explore. This test case aims to approximate scenarios such as architectural previewing or visiting a virtual museum.

Eight people participate in each experiment. The participants did not wear glasses during the experiments, since these affected the accuracy of pupil detection and we therefore restricted the experiments to participants that could experience VR normally without glasses.

In the first task, we explained the objective of the game and asked the participants to aim for a high score. In the second task, we asked them to explore the scenes and try to memorize some of their distinct features. The game task always preceded the exploration task. Before starting with either task, a short trial run ensured that the participants were comfortable and familiar with the objectives.

Figure 9 illustrates the process of the first experiment. Users first perform an initial calibration (~20 seconds). The calibration markers are immediately shown again, to give an estimate on the accuracy of the initial calibration process. Then, after 60 seconds of VR content, the accuracy of our drift estimation is similarly measured by displaying calibration markers again and comparing the gaze estimates to our results. These ~80 seconds sequences (VR content, followed by an accuracy measurement) are repeated a number of times (Table 1). Note that our drift estimation process is paused during the accuracy measurements and does not use data from them.

In the *adjustment experiment*, at a random time during one of the sequences during the 60 seconds of VR content, we prompt the user to adjust the headset, and record this moment in time. Users were asked to re-adjust the headset to a comfortable position to mimic real-world behaviour. The random time such that we record at least three accuracy measurements before and after

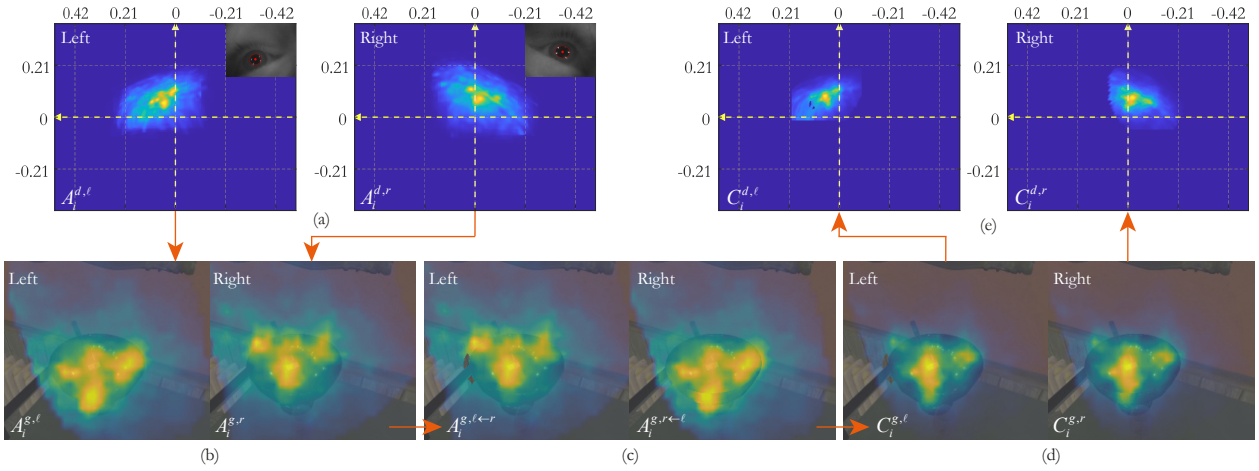


Fig. 7: We exploit stereo consistency by filtering the accumulation maps of the left and right eyes with each other. For example, the accumulation map $A_i^{d,r}$ is first transformed from drift vector space space (a) to gaze space (b). Maps are projected between the eyes using the camera and projection matrices and the depth buffer retained from the rendered images. Here, the reprojected accumulation map $A_i^{g,\ell\leftarrow r}$ (c) and the corresponding accumulation map (e.g, $A_i^{g,\ell}$) are merged into a combined map (d), $C_i^{g,\ell}$. The result is mapped back into drift-vector space (e). Instantaneous estimates are then drawn from the combined map $C_i^{d,\ell}$ instead of the original accumulation map ($A_i^{d,\ell}$).

Table 1: Overview of the user study experiments.

	Adjustment Experiment		Reseating Experiment	
	Game Task	Exploration Task	Game Task	Exploration Task
Scenes	Holodeck w. robots	San Miguel & Bistro	Holodeck w. robots	Bistro
Recorded sequences	≥ 10	≥ 10	4	4

the adjustment.

Each task lasts for at least ten sequences (when adjustments took place late (eighth sequence), an eleventh sequence was recorded to ensure that the sessions included at least three measurements after the adjustment). The exploration task switches scenes from the San Miguel scene to the Bistro scene after five sequences.

In the *reseating experiment*, participants perform the initial calibration (which is saved) and then immediately remove the headset. Participants are then asked to put on the headset again, at which point we load the saved calibration data and perform an accuracy test immediately. Following, we record four sequences as described for the first experiment. The second experiment only uses the Bistro scene in the exploration task.

Our user study was approved by the ethics assessment board of our institution, which follows the Declaration of Helsinki. Experiments were aborted if participants felt any discomfort (e.g., motion sickness). The recorded data is the basis of a two-fold analysis of our approach. We demonstrate that the proposed 2D drift-vector model can produce good correction results (Section 4.2) and we show that our on-the-fly estimation method can estimate high-quality 2D drift vectors (Section 4.3) .

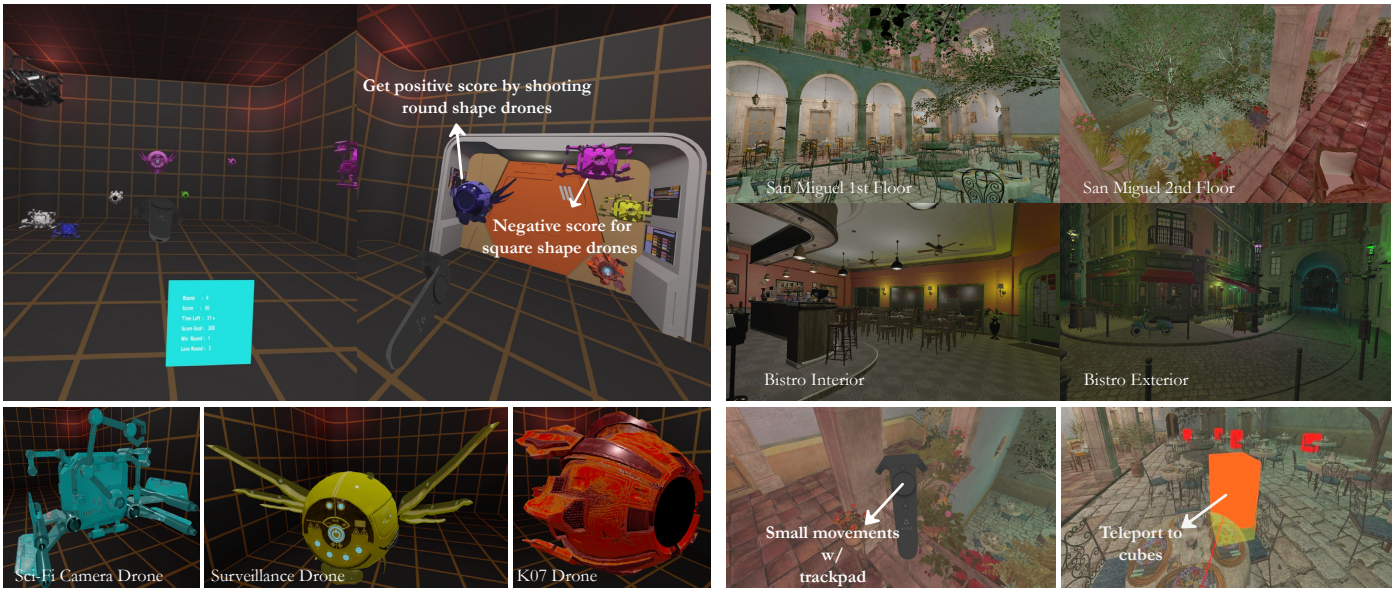
4.2. Analysis of the 2D drift vector model

Here, we verify our assumption that drift vectors can provide sufficient correction. We first establish an error baseline and perform an initial calibration immediately followed by another calibration step. By comparing the results of these two calibrations, we can estimate a base inaccuracy. Besides the average error E_{avg}, which is the average 2D distance between the estimated gaze positions and the corresponding marker, we also report E_{max}, the maximum 2D distance. For convenience, the distances are expressed in angles using the total field of view ($100° \times 110°$). Figure 10(a) shows the results of this initial test (due to the relatively short duration of these tests, we were able to collect data from 20 participants in total).

We then show the gaze estimation errors after a shift has occurred in Figure 10(b) and evaluate the accuracy of a gaze estimation involving an error-minimizing drift $\hat{\mathbf{d}}$ in Figure 10(c). For these two figures, we use all calibration moments during the user study. The error-minimizing drift vector $\hat{\mathbf{d}}$ for this verification is obtained using a brute-force search minimizing Equation 2. The calibration accuracy after a shift decreases significantly and the average error is around 10 degrees. However, the gaze estimation error can be corrected effectively by the error-minimizing drift vector. This ideal drift error correction shows comparable quality to the baseline; the worst case error using the error-minimizing drift vector is around 6°, and the average error is around 1.4°.

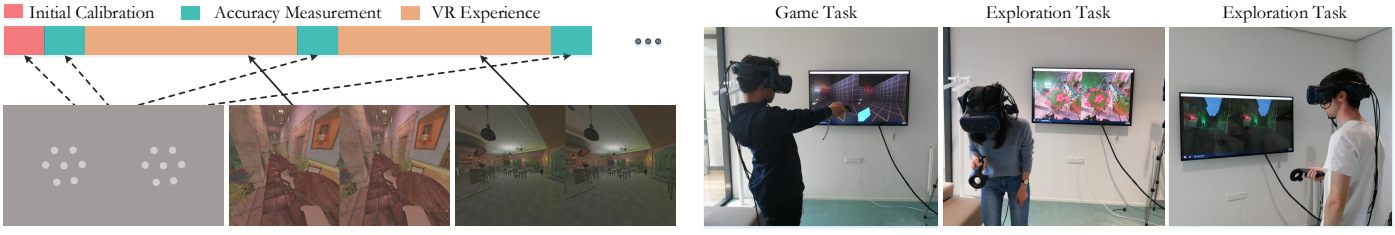
Figure 11 (top row) shows how the gaze estimation error after a shift and our ideal correction error change with drift magnitude. As expected, average and worst case errors are relatively constant with the error-minimizing correction, regardless of drift size. In comparison, the uncorrected gaze estimates show a linear relationship between the average gaze estimation error and drift size. This can be understood intuitively: without correction, the error is caused mainly by the drifts. The maximum error behaves worse than the average. We attribute this to the calibration method and its reliance on polynomial functions. The mapping becomes rapidly worse outside of the space spanned by the calibration points (as can be seen in Figure 1), so any outliers are amplified strongly.

The rightmost pair of the images in the bottom row of Figure 11 illustrates another potential source of errors. In general, a



(a) Game task (b) Exploration task

Fig. 8: (a) The game task. Participants are tasked with aiming at and shooting moving robotic drones. During the game, the users can see their scores in the cyan middle rectangle. The drones are randomly picked from three possible models (bottom row) and are tinted with a random color. (b) The exploration task. The exploration task takes place in two scenes: in the San Miguel 2.0 scene (top two images) and in the Amazon Lumberyard Bistro scene (middle two images). Users are free to explore both scenes, and move either physically or by using the controllers (either via teleporting to selected points or by using a button to move around as shown in the bottom images). The navigation cubes for position selection are only shown when users indicate that they want to move to a new location by pressing a button on the controller (this is a simplified version of the commonly used point and teleport locomotion technique [28]). The navigation cubes are shown 1.34s on average and frames with cubes account for ~20% of the total frames selected for processing. *The Holodeck scene (game task) and the San Miguel and Bistro scenes (exploration task) are from the Computer Graphics Archive [29]. Drone models (left to right) are: Sci-Fi Camera Drone by San Ozbulbul, Surveillance Drone by Tatiana Kunitsyna, and K07 Drone by KBM7 (the electronic version contains links to the model sources). Rendered models differ slightly from the originals due to exporting/importing limitations. Models are licensed under CC BY .*



(a) Experiment overview (b) User study in progress

Fig. 9: (a) Experiments start with a calibration step, which is immediately repeated to estimate the accuracy of the calibration. Accuracy measurements are repeated periodically during each experiment. (b) Between the measurements (where users only see the calibration markers), users perform activities related to one of the two tasks (game or exploration). Besides interaction via the HTC Vive Controllers, users can walk around physically (within limits of the room, which is approximately $2 \times 1.5\text{m}^2$).

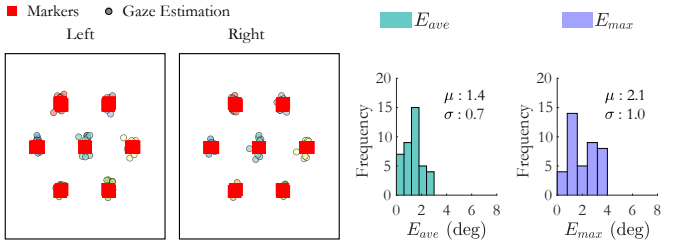
smaller eye image tends to result in larger errors, since the accuracy of the estimated pupil position decreases. In the rightmost pair of images, a large drift that places the pupil of the eye at the very edge of the eye-tracker image can be observed. In such cases, and especially at grazing angles, the pupil position may not be reported accurately.

Overall, these results show that a correction via a 2D drift vector performs well if the vector is correctly estimated.

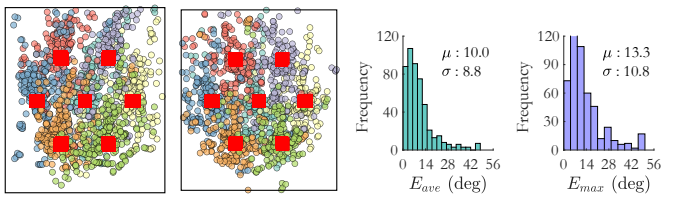
4.3. Analysis of the drift vector estimation

We estimate the 2D drift vectors on-the-fly from the rendered images, as described in Section 3.2 and saliency estimations are accumulated over time. From this accumulation, instantaneous estimates are extracted and subsequently filtered to derive a smoothed final estimate. Figure 12 displays complete timelines from two of our experiments, where we include the instantaneous estimates $\mathbf{d}_i$ (green circles) and the smoothed estimate $\bar{\mathbf{d}}_i$ (orange line). The error-minimizing drifts, which we can derive from our accuracy measurements, are included as well (green squares); the time periods when accuracy measurements take place are shown in a darker gray, during which our on-the-fly estimate is paused.

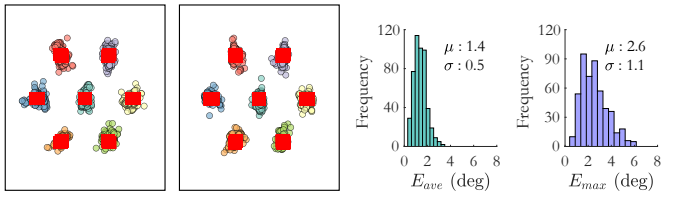
The first example (Figure 12, left) is from an adjustment experiment. The adjustment happens at around 400 seconds into the experiment (marked with a solid purple rectangle). It includes a brief period with noisy instantaneous estimates. These are filtered and do not affect the drift estimate significantly. Afterwards our drift vector estimate adjusts relatively quickly. We can see that the new estimated drift vector matches the drift from the next accuracy measurement quite well. The second



(a) Gaze estimation errors baseline



(b) Gaze estimation errors after a shift of the HMD



(c) Gaze estimation errors after corrected by an error-minimizing drift vector $\hat{\mathbf{d}}$

Fig. 10: Gaze estimation accuracy comparison. The left two scatter diagrams visualize the recorded gaze positions for each of the calibration markers (red points) shown during accuracy tests. The right two histograms show the distributions of both the average error and the maximum error (μ and σ denote average errors and standard deviation thereof). (a): The base accuracy, found by comparing the initial calibration with the results of an immediate accuracy measurement. (b): The accuracy without correction after a shift appears. (c): The accuracy achieved using the error-minimizing drift vector $\hat{\mathbf{d}}$ to correct the inaccurate results of (b).

example (Figure 12, right) shows a reseating experiment. Here, it takes about one minute before SalientGaze provides a good estimate of the drift vector. The initially noisy instantaneous estimates are aggressively filtered. This delays the convergence somewhat, but increases robustness.

Figure 13 summarizes the results from all experiments using SalientGaze with stereo: the top row shows the accuracy at the three accuracy measurements preceding the manual adjustment of the headset in the experiment. The middle row shows the three measurements following it. Note that some displacements are quite severe. In one case, one participant's eye data in the game task is not used in Figure 13(b) because the eye moved out of the view of the eye-tracking camera. The bottom row plots the accuracy of the first four measurements in the reseating experiment. The corresponding time spans in the examples in Figure 12 are marked with same-colored bars drawn at the top of the various plots. Left plots are for our game tasks, and the right columns for the exploration tasks.

Before the adjustment, we observe very few large drifts in our experiments. Consequently, the errors in the uncorrected case and with our method remain low. After the adjustment, SalientGaze generally provides gaze estimates with much lower errors. In one of the cases (Figure 13, middle row, right column, dark blue), the participant's manual adjustment was very close to the following accuracy measurement. In this case, SalientGaze was unable to converge to a new value in time, and instead recovers in time for the next measurement. The two outliers (two eyes of one participant) in the bottom row (right column, green circles) stem similarly from an unexpected slippage of the headset just before the accuracy test. Figure 14 explores this in more detail.

The timeline of Figure 14 begins at $t = 9$s after the user moves the headset; this is about one minute before the first accuracy test shown in Figure 13. By manually tracking the corners of the eyes, we can see that the headset shifts at around $t = 55$s. At this point, SalientGaze attempts to converge towards this drift. However, the headset shifts back already at about $t = 75$s, just a few seconds before the accuracy test is performed. This gives SalientGaze insufficient input to converge towards the new drift; SalientGaze does complete convergence only shortly after the accuracy test.

In consequence, we also added Figure 15, which takes this aspect into account. For all the measurements in Figure 13, we also compare to the drift estimation 30s after the calibration step. Implicitly, if there is a shift close to the calibration, we assume it will either happen before or after the calibration in a 30s interval, but not in both. As we can see, all the outliers disappear, which confirms the hypothesis that SalientGaze converges to the correct estimate. We therefore conclude that the outliers are actually introduced mostly by the timing of measuring the ground-truth drift vectors.

We summarize the average accuracies for all experiments (measurements in Figure 13 and measurements in Figure 15) in Table 2. SalientGaze performs overall better in the game task, with the very clearly salient objects. The exploration task takes place in scenes with a more uniform salience, which results in a higher standard deviation in Table 2, due to the absence of clear salient objects, which leads to a higher uncertainty. In the time before any adjustments, the accuracy with SalientGaze is slightly better compared to the uncorrected accuracy, which illustrates that it does not diverge. After an adjustment, the difference to the uncorrected case becomes obvious. The method can catch minute drifts and shifts, and compensate for these. SalientGaze is able to return to similar accuracy as before the adjustment, while the uncorrected gaze estimates show large errors. Reseating shows a similar pattern, where SalientGaze can recover to accuracies similar to the base case.

Exploiting stereo consistency improves results slightly versus providing just independent estimates, in particular in the exploration task. This is especially visible in the total sum of all errors, *SE* in Table 2, over the full data.

An advantage of SalientGaze is that performing gaze estimates is efficient. For the conversion from pupil to gaze space, we only need to find the pupil positions in the eye images, translate them by the current drift vector estimate, and then apply the mapping G. Regarding processing, only about 2 frames per second are evaluated for saliency by omitting frames based on the test for pupil movements. The latter only takes 1.5μs on average. The implementation provided by the authors of the saliency detection method [31] runs on the CPU, and takes around 45ms on average

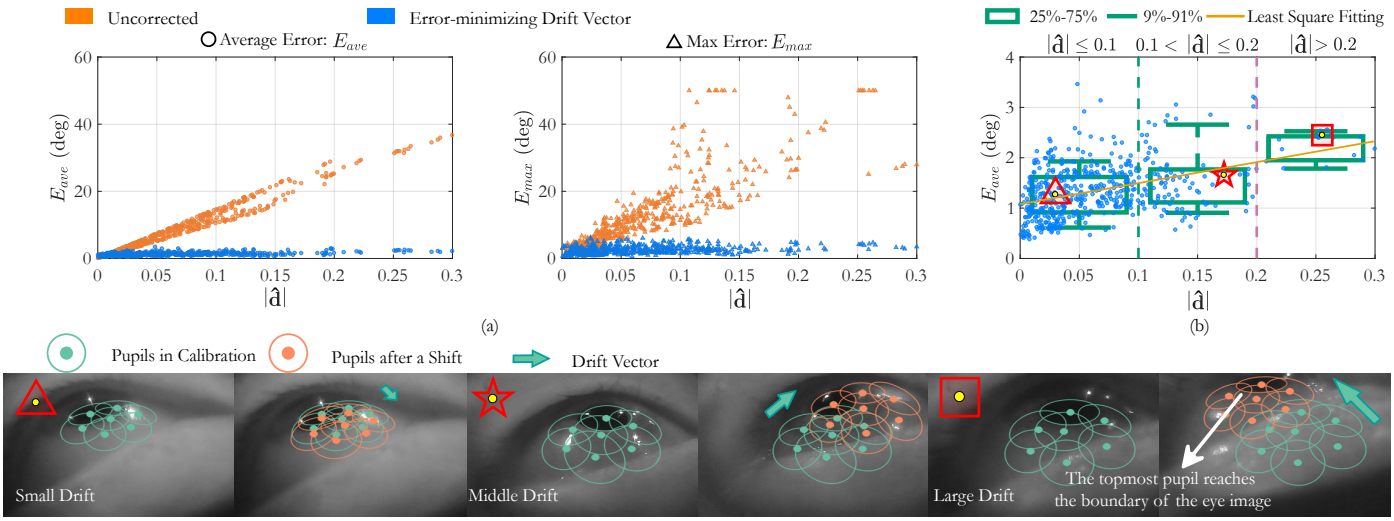


Fig. 11: Achievable accuracies. (a) Measured errors (average/max) are shown for different drift vector lengths. As expected, errors for the uncorrected case grow linearly with increasing drift vector lengths (which indicate increasing shifts). Using the error-minimizing drift vector for correction results in a low measured errors in the gaze estimation across the board. (b) A closer look at how the average error E_{ave} changes with the magnitude of the error-minimizing drift vector magnitude $|\hat{\mathbf{a}}|$. Sample eye images for select cases in different regimes of drift vectors (cases marked with the corresponding symbol in (b)) are shown in the bottom row. The rightmost eye image pair is the most problematic one: it combines a large drift that shifts the eye very close to the border of the image, where accurate pupil detection also becomes problematic.

Table 2: Average accuracies from our experiments. Uncorr.: the initial calibration accuracy; Ours (M): our correction only considers mono saliency; Ours (S): our correction with stereo information considered; adj.: adjustment; reseat.: reseating. The first measurement after adjustment is not taken when performing the statistic on the accuracy of our method because the adjustment is very close to the accuracy measurement. For a fair comparison during the accuracy statistic, the first measurement in the data before adjustments is also removed, which means that the measurements in the dashed rectangles of Figure 13 and Figure 15 are not considered. Avg. is the average, Std. is the standard deviation and SE is the sum of errors.

(a) Statistics of the measurements in Figure 13.

Game Task	Before adj.		After adj.		After reseat.		
	Avg.	Std.	Avg.	Std.	Avg.	Std.	SE
Uncorr.	3.90	2.47	13.71	11.56	12.24	8.13	1319.80
Ours (M)	3.25	1.38	3.04	1.21	3.21	1.20	400.74
Ours (S)	2.93	1.05	3.03	1.19	3.18	1.33	388.46

Explor. Task	Before adj.		After adj.		After reseat.		
	Avg.	Std.	Avg.	Std.	Avg.	Std.	SE
Uncorr.	4.84	2.71	13.63	9.86	11.64	5.55	1335.99
Ours (M)	3.86	2.63	4.10	2.00	4.63	3.69	551.04
Ours (S)	3.87	2.93	3.86	2.04	3.99	2.71	502.87

(b) Statistics of the measurements in Figure 15.

Game Task	Before adj.		After adj.		After reseat.		
	Avg.	Std.	Avg.	Std.	Avg.	Std.	SE
Ours (M)	3.00	1.30	2.68	0.99	2.95	1.00	365.35
Ours (S)	2.70	1.05	2.49	1.05	2.83	1.10	342.22

Explor. Task	Before adj.		After adj.		After reseat.		
	Avg.	Std.	Avg.	Std.	Avg.	Std.	SE
Ours (M)	3.23	2.20	3.45	2.10	3.45	2.11	434.22
Ours (S)	3.06	1.98	3.18	1.85	2.99	1.65	390.97

for each of the two 288×299 pixel images. The remaining parts of our pipeline execute on the GPU, but are not fully optimized for performance. Nevertheless, the total computation time from saliency to an updated drift vector takes on average only 2.2ms in total for both eyes without stereo and 4.2ms with stereo.

The average accuracy of our gaze correction is around 3° for the game scenario and 4° for the exploration scenario. The achieved accuracy is comparable to that of related contemporary methods. To place such a performance in context, the authors of the machine learning-based NVGaze [9] report an average accuracy of 3.5°. The SynthesEye-base approach [16] reports a 7.9° accuracy on average, with some worst-case error above 10°. Geometric models perform similarly, e.g., the work by Stengel et al. [20] shows an accuracy of 0.5° to 3.5° but it requires a precise model of the HMD, making it device dependent. While a direct comparison is difficult, SalientGaze avoids the mentioned pitfalls, as well as the lengthy training processes of machine learning-based approaches, while achieving comparable results.

SalientGaze relies heavily on saliency detection. We briefly investigated performance with other top methods identified by the MIT saliency benchmark [32]. Using the deep-learning based Saliency Attentive Model (SAM-ResNet) [40] improves accuracy slightly: total errors (*SE* in Table 2) are reduced by about 4.6% in the game task and change less than 1% in the exploration task. For development, we preferred the boolean map saliency method. Using a more traditional vision-based algorithm avoided many dependencies on large software packages used for learning-based methods. Nevertheless, users of SalientGaze can choose any saliency detection method according to their needs and preferences. We also experimented with normalization of the saliency map. In essence, normalization would remap values of each S_i such that the range is $[0, 1]$ using the minimum and maximum values found in that frame. Total errors increase slightly in the exploration task (~2% for the Boolean Map Saliency and ~9% for the Saliency Attentive Model) and change less than 1% in the game task across the board. We thus recommend working with non-normalized saliency maps, that is, with the maps as returned by the corresponding methods.

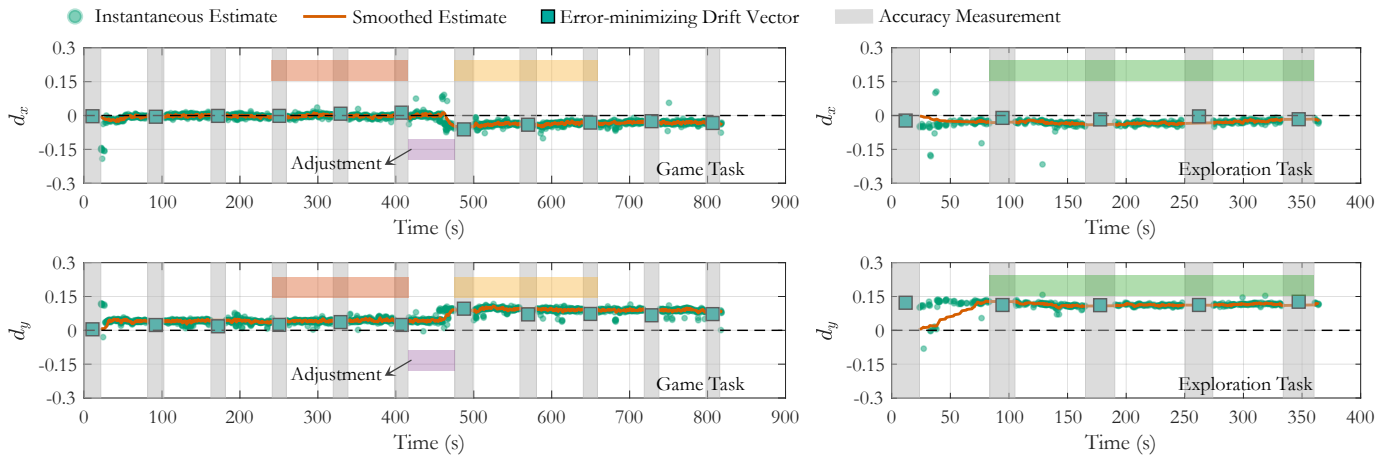
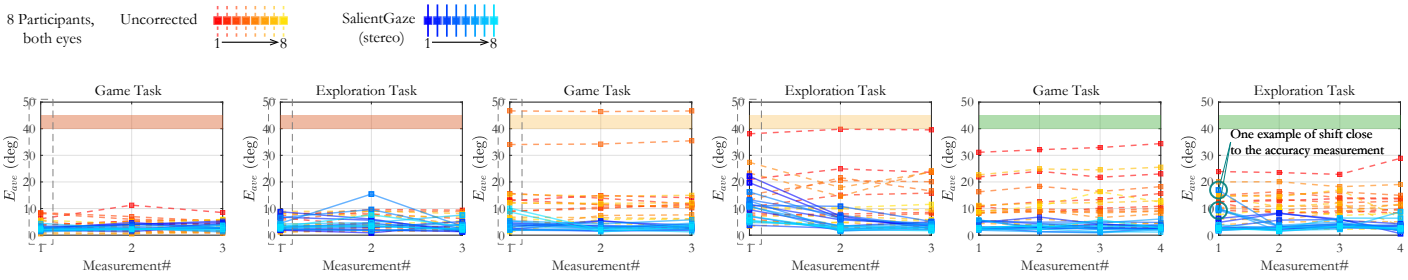


Fig. 12: Two drift estimation examples from our two experiments. In the adjustment experiment, since users adjust the headset at different time, we choose three measurements before the adjustment (red filled ractangles) and three measurements after the adjustment (yellow filled rectangles) to compare how our algorithm behaves among different participants. In the headset reseating experiment, four measurements (green filled rectangles) are chosen, one minute after users reseat the headset. The colored rectangles indicate the measurements from this experiment that contribute to the summarized results shown in Figure 13 (see matching color bars).



(a) Before adjustment (b) After adjustment (c) After reseating the headset

Fig. 13: Results from the experiments. In each pair of plots, the left plot presents measurements for the game task, and the right plot for the exploration task. The color of the top rectangles matches with the measurements highlighted by the same-colored rectangles in Figure 12. (a) SalientGaze shows similar results as the initial calibration when no large drift occurs. (b) SalientGaze still shows good results after users adjust the headset, whereas uncorrected accuracy suffers. (c) Likewise, SalientGaze recovers even when users take off and don the headset again. The measurements marked with green circles are studied further in Figure 14, as in this case a drift occurred just before the accuracy measurement. The measurements in dashed rectangles are omitted from the results in table 2, as discussed there.

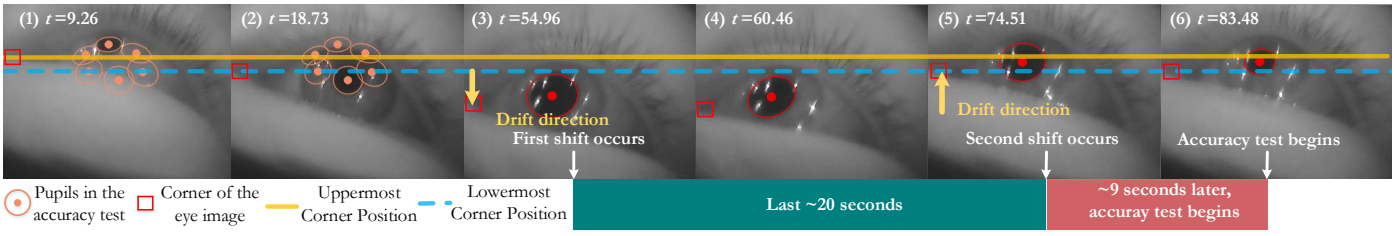
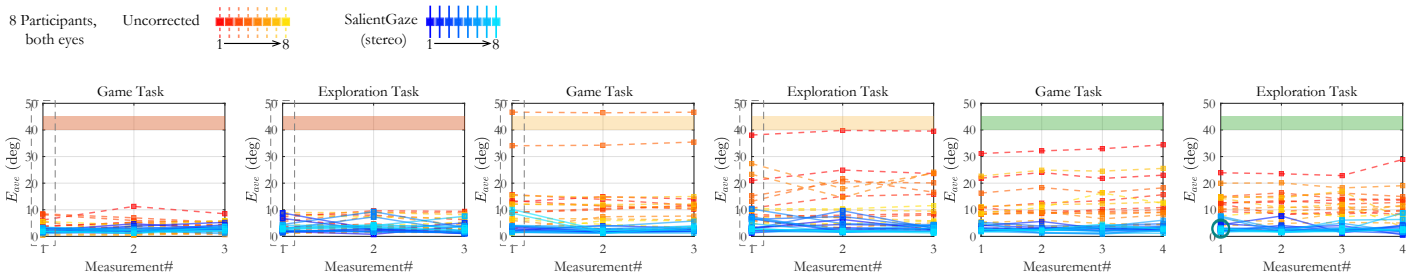


Fig. 14: Example of an unexpected shift that occurs close to one of our accuracy measurements. This causes a large error, as SalientGaze has not converged to a new drift vector at the time of the accuracy measurement. A first shift of the HMD occurs around t = 55s. SalientGaze starts converging towards this shift. However, a second shift occurs at t = 75s, approximately ten seconds before the accuracy test. SalientGaze does not react quickly enough to reflect this in time for the accuracy test (but converges afterwards).

5. Conclusions and future work

In real-life conditions, a VR headset can shift and move relative to the user's head, both naturally and through active intervention by the user. In our study, we use artificial shifts to simulate naturally occurring ones. The latter are caused by rapid head movements, but also by, e.g., stepping on cables attached to the HMD. Natural shifts are often smaller in nature than our artificial ones but still benefit strongly from a correction. For gaze estimation with near-eye cameras attached to the headset, shifts cause problems for methods relying on calibration. First, we showed that a wide spectrum of shifts can accurately be modelled with a simple 2D drift vector in pupil space. Next, we proposed a simple on-the-fly method to estimate such 2D drift vectors from saliency in the shown VR content. We demonstrated SalientGaze in a realistic setting with two complex tasks and under relatively extreme conditions, i.e., when users adjust or even completely



(a) Before adjustment (b) After adjustment (c) After reseating the headset

Fig. 15: Use the drift estimation 30 seconds later to correct all the measurements in Figure 13 if the drift vector 30s later shows lower error than the current drift vector. We can see that all the outliers disappear. This shows that SalientGaze does give a correct drift estimation later on for measurements that show large error in Figure 13.

remove and redo the VR headset. SalientGaze reduces errors significantly compared to an uncorrected situation.

A drawback is the heavy reliance on saliency information. To test our algorithm in a difficult scenario, the 3D scene in our exploration task lacks very clear salient areas, but it results in less information to be used by our estimator. One interesting approach for future work would be artificial intervention, injecting salient objects/content at key times, potentially involving subtle cues [41]. Also, if one can identify salient objects ahead of time, saliency computations might be avoided altogether. This would be possible in, for example, our game tasks, where we know that the target robots are the key salient objects, or when the user is interacting with an interface similar to the method proposed by Sidenmark [27].

SalientGaze is capable of providing results with accuracies around 3° in scenarios with clear salient objects and 4° on average in others. This is already sufficient for many practical applications such as foveated rendering, which is considered a key technology for lowering rendering costs in the future.

Acknowledgments

We want to thank Dr. L. Scandolo for advice on the code, reading this paper, and giving suggestions. We also want to say thank you for Dr. M. Stengel for initial discussions and his C++ application coding framework.

Funding: This work was funded by the China Scholarship Council (No. 201603170298); the Swiss National Science Foundation (Advanced Postdoc Mobility Project 174321); and the VIDI grant NextView funded by NWO Vernieuwingsimpuls.

References

[1] Patney, A, Salvi, M, Kim, J, Kaplanyan, A, Wyman, C, Benty, N, et al. Towards foveated rendering for gaze-tracked Virtual Reality. ACM Trans Graph 2016;35(6). doi:10.1145/2980179.2980246.

[2] Weier, M, Stengel, M, Roth, T, Didyk, P, Eisemann, E, Eisemann, M, et al. Perception-driven accelerated rendering. In: Computer Graphics Forum (Proceedings of Eurographics); vol. 36. Eurographics; Oxford: Blackwell-Wiley; 2017,URL: http://graphics.tudelft.nl/Publications-new/2017/WSRDEEGHKMMS17.

[3] Albert, R, Patney, A, Luebke, D, Kim, J. Latency requirements for foveated rendering in Virtual Reality. ACM Trans Appl Percept 2017;14(4). doi:10.1145/3127589.

[4] Santis, AD, Iacoviello, D. Robust real time eye tracking for computer interface for disabled people. Computer Methods and Programs in Biomedicine 2009;96(1). doi:10.1016/j.cmpb.2009.03.010.

[5] Song, G, Cai, J, Cham, TJ, Zheng, J, Zhang, J, Fuchs, H. Real-time 3d face-eye performance capture of a person wearing VR headset. In: Proceedings of the 26th ACM International Conference on Multimedia. MM '18; New York, USA: ACM; 2018,doi:10.1145/3240508.3240570.

[6] Durand, F, Dorsey, J. Interactive tone mapping. In: Péroche, B, Rushmeier, H, editors. Rendering Techniques 2000. Vienna: Springer Vienna; 2000,doi:10.1007/978-3-7091-6303-0_20.

[7] Chandon, P, Hutchinson, J, Bradlow, E, Young, SH. Measuring the value of point-of-purchase marketing with commercial eye-tracking data. INSEAD Business School Research Paper 2006;(2007/22). doi:10.2139/ssrn.1032162.

[8] Arabadzhiyska, E, Tursun, OT, Myszkowski, K, Seidel, HP, Didyk, P. Saccade landing position prediction for gaze-contingent rendering. ACM Transactions on Graphics (Proc SIGGRAPH) 2017;36(4). doi:10.1145/3072959.3073642.

[9] Kim, J, Stengel, M, Majercik, A, De Mello, S, Dunn, D, Laine, S, et al. NVGaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In: Proceedings of the CHI Conference on Human Factors in Computing Systems. 2019,doi:10.1145/3290605.3300780.

[10] Sugano, Y, Matsushita, Y, Sato, Y. Calibration-free gaze sensing using saliency maps. In: IEEE Conference on Computer Vision and Pattern Recognition. 2010,doi:10.1109/CVPR.2010.5539984.

[11] Sugano, Y, Matsushita, Y, Sato, Y. Appearance-based gaze estimation using Visual Saliency. IEEE transactions on pattern analysis and machine intelligence 2013;35(2). doi:10.1109/TPAMI.2012.101.

[12] Sugano, Y, Bulling, A. Self-calibrating head-mounted eye trackers using egocentric Visual Saliency. In: Proceedings of the Annual ACM Symposium on User Interface Software; Technology. UIST '15; 2015,doi:10.1145/2807442.2807445.

[13] Kar, A, Corcoran, P. A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms. CoRR 2017;abs/1708.01817. URL: http://arxiv.org/abs/1708.01817. arXiv:1708.01817.

[14] Hansen, DW, Ji, Q. In the eye of the beholder: A survey of models for eyes and gaze. IEEE transactions on pattern analysis and machine intelligence 2010;32(3). doi:10.1109/TPAMI.2009.30.

[15] Krafka, K, Khosla, A, Kellnhofer, P, Kannan, H, Bhandarkar, S, Matusik, W, et al. Eye tracking for everyone. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016, p. 2176–2184. doi:10.1109/CVPR.2016.239.

[16] Wood, E, Baltruaitis, T, Zhang, X, Sugano, Y, Robinson, P, Bulling, A. Rendering of eyes for eye-shape registration and gaze estimation. In: Proceedings of the 2015 IEEE International Conference on Computer Vision. ICCV '15; USA: IEEE Computer Society; 2015,doi:10.1109/ICCV.2015.428.

[17] Soccini, AM. Gaze estimation based on head movements in virtual reality applications using deep learning. In: 2017 IEEE Virtual Reality (VR). 2017,doi:10.1109/VR.2017.7892352.

[18] Xu, Y, Dong, Y, Wu, J, Sun, Z, Shi, Z, Yu, J, et al. Gaze prediction in dynamic 360° immersive videos. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, p. 5333–5342. doi:10.1109/CVPR.2018.00559.

[19] Guestrin, ED, Eizenman, M. General theory of remote gaze estimation using the pupil center and corneal reflections. IEEE Transactions on Biomedical Engineering 2006;53(6). doi:10.1109/TBME.2005.863952.

[20] Stengel, M, Grogorick, S, Eisemann, M, Eisemann, E, Magnor, MA. An affordable solution for binocular eye tracking and calibration in head-mounted displays. In: Proceedings of the ACM International Conference on Multimedia. 2015,doi:10.1145/2733373.2806265.

[21] Kassner, M, Patera, W, Bulling, A. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In: Adjunct Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing. UbiComp '14 Adjunct; 2014,doi:10.1145/2638728.2641695.

[22] Itti, L, Koch, C, Niebur, E. A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998;20(11). doi:10.1109/34.730558.

[23] Borji, A, Itti, L. State-of-the-art in visual attention modeling. IEEE Transactions on Pattern Analysis and Machine Intelligence 2013;35(1). doi:10.1109/TPAMI.2012.89.

[24] Chen, J, Ji, Q. Probabilistic gaze estimation without active personal calibration. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society; 2011,doi:10.1109/CVPR.2011.5995675.

[25] Perra, D, Gupta, RK, Frahm, J. Adaptive eye-camera calibration for head-worn devices. In: IEEE Conference on Computer Vision and Pattern Recognition. 2015,doi:10.1109/CVPR.2015.7299042.

[26] Tripathi, S, Guenter, B. A statistical approach to continuous self-calibrating eye gaze tracking for head-mounted Virtual Reality systems. In: IEEE Winter Conference on Applications of Computer Vision. 2017,URL: http://arxiv.org/abs/1612.06919.

[27] Sidenmark, L. Immersive eye tracking calibration in virtual reality using interactions with in-game objects. Master's thesis; KTH, School of Computer Science and Communication; 2017. URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1111978&dswid=8411.

[28] Bozgeyikli, E, Raij, A, Katkoori, S, Dubey, R. Point & teleport locomotion technique for virtual reality. In: Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play. New York, USA: Association for Computing Machinery; 2016,doi:10.1145/2967934.2968105.

[29] McGuire, M. Computer graphics archive. 2017. URL: https://casual-effects.com/data.

[30] Salvucci, DD, Goldberg, JH. Identifying fixations and saccades in eye-tracking protocols. In: Proceedings of the 2000 Symposium on Eye Tracking Research & Applications. 2000,doi:10.1145/355017.355028.

[31] Zhang, J, Sclaroff, S. Exploiting surroundedness for saliency detection: A boolean map approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 2016;38(5). doi:10.1109/TPAMI.2015.2473844.

[32] Bylinskii, Z, Judd, T, Borji, A, Itti, L, Durand, F, Oliva, A, et al. MIT saliency benchmark. 2019. URL: http://saliency.mit.edu/results_mit300.html.

[33] Bylinskii, Z, Judd, T, Oliva, A, Torralba, A, Durand, F. What do different evaluation metrics tell us about saliency models? IEEE Transactions on Pattern Analysis and Machine Intelligence 2019;41(3). doi:10.1109/TPAMI.2018.2815601.

[34] Judd, T, Durand, F, Torralba, A. A benchmark of computational models of saliency to predict human fixations. In: MIT Technical Report. 2012,URL: http://hdl.handle.net/1721.1/68590.

[35] Sitzmann, V, Serrano, A, Pavel, A, Agrawala, M, Gutierrez, D, Masia, B, et al. Saliency in VR: How do people explore virtual environments? IEEE Transactions on Visualization and Computer Graphics 2018;24(4). doi:10.1109/TVCG.2018.2793599.

[36] Scherzer, D, Yang, L, Mattausch, O, Nehab, D, Sander, PV, Wimmer, M, et al. A Survey on Temporal Coherence Methods in Real-Time Rendering. In: John, N, Wyvill, B, editors. Eurographics 2011 - State of the Art Reports. The Eurographics Association; 2011,doi:10.2312/EG2011/stars/101-126.

[37] Zhang, J, Green, S. Introducing stereo shading reprojection for Unity. 2017. URL: https://developer.oculus.com/blog/introducing-stereo-shading-reprojection-for-unity/.

[38] Beat saber. 2019. URL: https://beatsaber.com/.

[39] Space pirate trainer. 2016. URL: https://www.spacepiratetrainer.com/.

[40] Cornia, M, Baraldi, L, Serra, G, Cucchiara, R. Predicting human eye fixations via an LSTM-based saliency attentive model. IEEE Transactions on Image Processing 2018;27(10):5142–5154. doi:10.1109/TIP.2018.2851672.

[41] Grogorick, S, Stengel, M, Eisemann, E, Magnor, M. Subtle gaze guidance for immersive environments. In: Douglas W. Cunningham, MB, editor. Proc. ACM Symposium on Applied Perception (SAP), ACM. 2017,URL: http://graphics.tudelft.nl/Publications-new/2017/GSEM17.