Gaze Mapping for Immersive Virtual Environments based on Image Retrieval

Tim Rolff [1,2], Frank Steinicke [1] and Simone Frintrop [2]

[1] *Universität Hamburg, Hamburg, Germany, Department of Human-Computer Interaction*
[2] *Universität Hamburg, Hamburg, Germany, Department of Computer Vision*

Correspondence*:
Tim.Rolff@uni-hamburg.de

ABSTRACT

In this paper, we introduce a novel gaze mapping approach for free viewing conditions in dynamic immersive virtual environments (VEs), which projects recorded eye fixation data of users, who viewed the VE from different perspectives, to the current view. This generates eye fixation maps, which can serve as ground truth for training machine learning (ML) models to predict saliency and the user's gaze in immersive virtual reality (VR) environments. We use a flexible image retrieval approach based on SIFT features, which can also map the gaze under strong viewpoint changes and dynamic changes. A vocabulary tree enables to scale to the large amounts of data with typically several hundred thousand frames and a homography transform re-projects the fixations to the current view.

To evaluate our approach, we measure the predictive quality of our eye fixation maps to model the gaze of the current user and compare our maps to computer-generated saliency maps on the *DGaze* and the *Saliency in VR* datasets. The results show that our method often outperform these saliency predictors. However, in contrast to these methods, our approach collects real fixations from human observers, and can thus serve to estimate ground truth fixation maps in dynamic VR environments, which can be used to train and evaluate gaze predictors.

Keywords: Gaze Mapping, Fixation Mapping, Free Viewing Environment, Eye Fixation Maps, Saliency, Gaze Re-projection, Ground Truth Fixation Estimation

1 INTRODUCTION

Rendering realistic virtual environments (VEs) in immersive virtual reality (VR) requires an enormous amount of computational power (Kanter, 2015). This is due to several VR-specific requirements, in particular, the demand for stereoscopic rendering with a high number of frames per second, and low end-to-end latency. Furthermore, recent standalone head-mounted displays (HMD) feature only limited CPU/GPU performance, which requires additional optimization (Hosny et al., 2020). However, providing high-fidelity VEs might not be necessary, as the human visual system can be divided into a foveal region, covering only a small visual angle of the field of view with sharp vision, and a surrounding region lacking sharpness (Holmqvist et al., 2011). This trait of human physiology can be exploited through *foveated rendering* (Patney et al., 2016). Here, frames are generated depending on the current gaze position of a user, which requires to capture the current fixation position on the display through an eye-tracker. Then, only pixels projected inside the foveal region are rendered with the highest resolution and quality, whereas

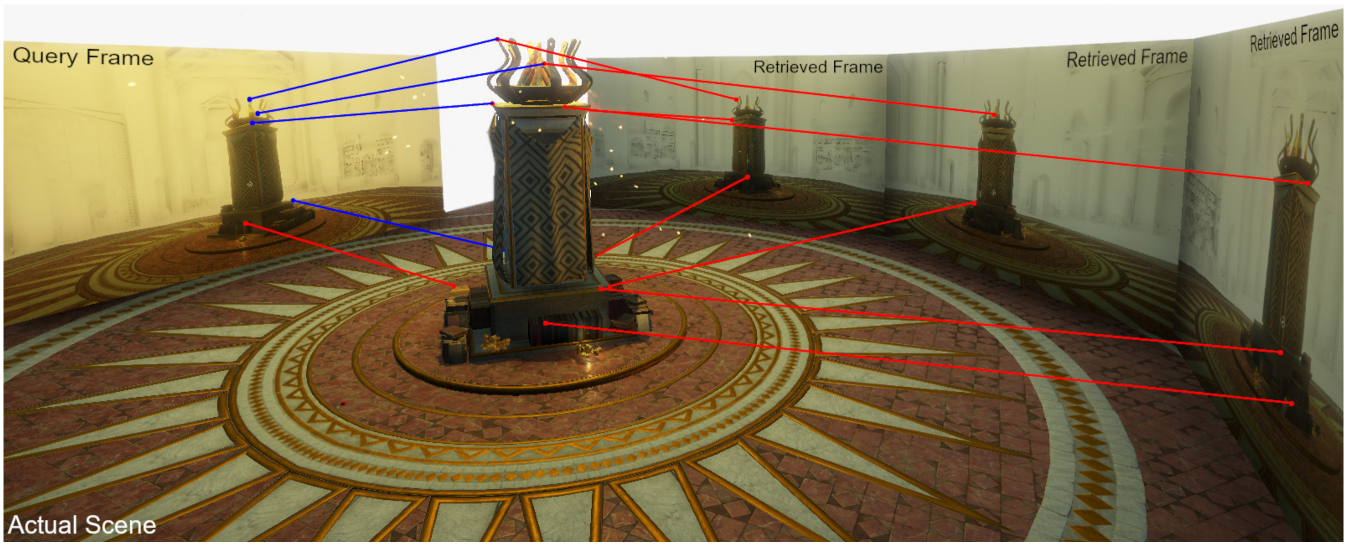

Figure 1. Visualization of our gaze mapping: For a *Query Frame*, similar frames are retrieved from the database *Retrieved Frames*, and captured gaze points (red dots) from other users are projected to the current frame (blue lines).

the remaining pixels outside this region are rendered with reduced resolution and quality. However, this solution requires a sufficiently fast eye-tracker, which provides fixation locations with low latency to the graphics processor. Otherwise, Albert et al. (2017) noted that depending on the latency, users might notice artifacts of the applied foveated rendering technique. Furthermore, a recent study by Stein et al. (2021) showed that some customer and professional HMDs still do not achieve latencies low enough to match the latency thresholds found by Albert et al. (2017). To further reduce the latency required to detect the gaze, gaze forecasting or gaze prediction (Nakashima et al., 2015; Huang et al., 2018; Hu et al., 2020, 2021) by calculating future gaze positions given the past gaze and the visual stimulus.

Developing a gaze predictor based on machine learning (ML) requires ground-truth eye fixations that allow to evaluate and optionally train such a system. Typically, eye fixation maps are acquired through capturing the gaze of multiple users viewing an image or video, and collecting the fixations (Bylinskii et al., 2020; Jiang et al., 2015; Mathe and Sminchisescu, 2012). In VR, an omnidirectional scene can be used (Sitzmann et al., 2018). This usually requires the users to maintain a predetermined viewpoint, often through fixation of the head (Holmqvist et al., 2011) or a fixed viewing position in VR (Sitzmann et al., 2018). However, in VR environments with free egocentric viewing conditions, generating such fixation maps is challenging, as users are allowed to look everywhere, even at places which other users might not have seen before. Hence, it is unlikely that two users will see the same visual stimulus, since this would only be possible if both participants observe the scene from the exact same viewpoint at any given time. Furthermore, in dynamic scenes, which can be changed, for example, by the users' actions, all users would need to go through the same alterations to capture the same viewpoint. This might not always be possible, as the state of the environment could be altered through a random process as well. Another challenge when capturing egocentric gaze in 3D arises due to different behavior when observing a 2D compared to egocentric 3D visual stimuli. In particular, egocentric gaze in 3D is more center biased when observing a 3D scene (Celikcan et al., 2020). Similar results are found by Foulsham et al. (2011) when walking in real environments, in which gaze is also more centered in egocentric vision.

A solution to project eye fixations from other users to the current view is *gaze mapping* (MacInnes et al., 2018a,b; Kraus et al., 2019; De Tommaso and Wykowska, 2019). Here, the captured gaze of a video of one user gets mapped onto a target image using a projection technique, such as a homography or perspective projection. To generate such a mapping, these methods usually apply standard feature matching techniques such as SIFT descriptor matching (Lowe, 2004) to align image regions from different viewpoints. While this approach is suitable to match to a single frames, it does not scale to our setting: we need to project all fixations from all other users to the current scene. Matching hundreds or even thousands of keypoints per frame of a typical VR datasets (e.g., *DGaze* (Hu et al., 2020)), which contains more than one million frames, would require over a 1000 billion matches.

Therefore, we propose an approach to extend gaze mapping to large VR environments based on image retrieval techniques. The purpose of our algorithm is to find frames, which correspond to the current view of the user, and map the eye fixations of the other users to the current view, schematically shown in Fig. 1 . As in other gaze mapping approaches, we compute SIFT feature descriptors to represent the image regions. However, since we need to compute descriptors for videos of all users in a dataset, we need a suitable data structure to represent the descriptors and to quickly find similar ones. For this, we build a *vocabulary tree*, as used in image retrieval approaches (Nister and Stewenius, 2006), which stores the descriptors in a tree structure. The tree is built by hierarchical k-means clustering, which quantizes the features into visual words, similar to a bag of words approach. The tree structure enables an efficient search for similar SIFT descriptors and thus to quickly find image regions, which show the same scene as the current frame from different viewpoints. Once a match is found, the corresponding eye fixations are projected to the current frame using a projective transformation. Finally, we generate an eye fixation map from all projected fixations, which can serve as ground truth for training and evaluating gaze predictors. The generated ground truth data that is the output of our approach along with the videos can then be utilized for the training of such ML gaze and saliency predictors. Figure 2 shows some of the fixation maps generated with our algorithm.

To summarize, our work proposes the following contributions:

- We provide a novel gaze mapping method for the generation of eye-fixation maps from free viewing video data containing correspondingly captured gaze points.
- We propose a framework that can be used in conjunction with the proposed algorithm to process image/video data containing several hundred thousand frames.
- We evaluate the predictive quality of our approach to model the gaze of the current user and compare it to computer-generated saliency maps like Boolean Map Saliency (Zhang and Sclaroff, 2013), Minimum Barrier Salient Object Detection (Zhang et al., 2015), VOCUS2 (Frintrop et al., 2015) and Salient Attentive Model (Cornia et al., 2018).

The remainder of this article is structured as follows. We will first introduce related work in Section 2, and provide background information about psychological and computer vision concepts. Afterwards, we will explain our method in Section 3 along with a short summary on assumptions made about data. Section 4 will then show the results of our proposed algorithm on the selected datasets. At last, we will conclude our work in Section 5.

2 RELATED WORK

2.1 Gaze Mapping

Gaze mapping describes the process of mapping someone's gaze from an input video frame or image onto a target image. This method partially solves the view dependency when observing a scene from different viewpoints through a mobile eye-tracker by projecting the captured gaze of one user onto a target image/frame of another user using computer vision algorithms. Most gaze mapping systems track a predefined area of interest (AOI) or project the captured gaze onto a target image.

Gaze mapping has been mostly used as a tool for the analysis of human gaze in specific scenarios. For example, Kurzhals et al. (2016); Benjamins et al. (2018); De Tommaso and Wykowska (2019) utilize gaze mapping to derive semantic interpretations of human viewing behavior during experiments by tracking predefined AOIs. These methods utilize image descriptors to match features of the provided target with the input frame for image re-identification the AOI. Other approaches, like the ones proposed by De Tommaso and Wykowska (2019); Pfeiffer et al. (2016); MacInnes et al. (2018a,b), directly map gaze points onto an initially provided static image. A slightly different approach by Pfeiffer et al. (2016) is also applicable for VR scenarios, as it tries to estimate the 3D position of a known 3D scene. However, it also requires information about the position and orientation towards the visualized content to be known beforehand.

2.2 Saliency & Visual Attention

In human vision, gaze is controlled by mechanisms of selective attention, which focus the gaze on regions of potential interest (Pashler, 1999). Visual attention can be computationally modelled by saliency methods, which compute saliency maps and can be used as gaze prediction. Earlier computational saliency predictors such as (Itti et al., 1998) model human attention through the Feature Integration Theory (Treisman and Gelade, 1980; Treisman and Kanwisher, 1998). This theory states that visual features are registered in parallel early in the visual process, while objects are formed in later stages from the collected features. This final step requiring active attention. Based on this theory (Koch and Ullman, 1987) formulated the concept of a saliency map. A saliency map is used to pool all features into a global map, where the attention is then allocated to the location with the highest activity. Later, approaches (Itti et al., 1998; Frintrop et al., 2010; Zhang and Sclaroff, 2013; Zhang et al., 2015) use handcrafted features such as intensity, contrast, or color to predict these saliency maps. More recent approaches like (Kummerer et al., 2017; Cornia et al., 2018; Che et al., 2019; Droste et al., 2020) utilize deep learning to predict the saliency map without extracting additional image features.

2.3 Image Retrieval

Image retrieval describes the process of retrieving similar images from a larger database (Nister and Stewenius, 2006; Uriza et al., 2018). Here, we base our definition on Uriza et al. (2018) who define *similar images* as images containing the same object under different viewing conditions such as scale, rotation, illumination. Then, given an arbitrary input image, the retrieval system will return multiple different images showing the same content as the query image.

One way to implement such a system is by using local feature detectors such as SIFT (Lowe, 2004), SURF (Bay et al., 2008) or ORB (Rublee et al., 2011). These methods calculate specific keypoints along with feature descriptors, which can be used to re-identify objects. Given a set of key points and feature descriptors for two images, it is possible to match both images by nearest neighbor search in feature space Lowe (2004).

One of the main challenges of image retrieval is scaling well with the number of images. Exhaustively matching all images in the database with the input would be infeasible for most applications. As a solution

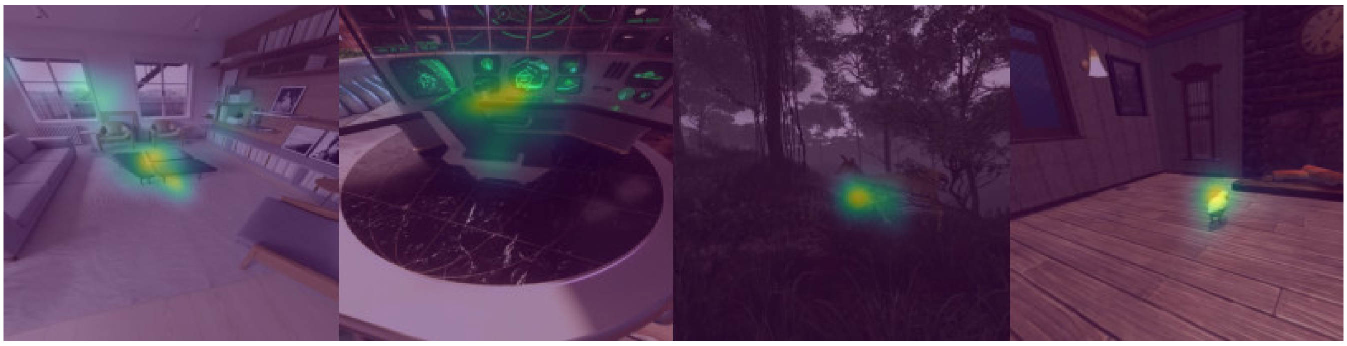

Figure 2. Eye fixation maps generated through our algorithm overlaid on-top of the actual image. The maps are generated by applying a Gaussian blur over the fixations projected through our algorithm. Section 3.2 explains the details on our algorithm. The images are from the *Saliency in VR* (Sitzmann et al., 2018) and the *DGaze* (Hu et al., 2020) datasets.

to this problem, Nister and Stewenius (2006) proposed the vocabulary tree data structure that scales well with the number of images. This data structure stores all images as vocables, similar to a bag of words approach. As input data, the extracted local features of a training dataset are quantized into visual words using a clustering algorithm such as k-means are used. Applying the quantization repeatedly in a hierarchical manner constructs the tree, which enables an efficient search for similar SIFT descriptors.

3 EYE-FIXATION MAP GENERATION

In this section, we will introduce our method for generating eye-fixations maps by gaze mapping in VR environments. To explain our algorithm, we will briefly explain some key-concepts that are used and afterwards describe the pipeline itself.

3.1 Assumptions

To generate fixation maps, our framework relies on the assumption that the gaze data from several users was acquired in the same physical or virtual space. This means the data consists of egocentric, first person video data along with the gaze locations. This data capturing is not only restricted to VEs through an HMD, but can be done either by using a wearable eye-tracker in real-world environments. To work properly, our algorithm requires that the captured videos contain the same scenes from multiple viewpoints. Otherwise, the algorithm will fall back to user-based fixation maps that only capture the gaze of the currently processed user. Therefore, the more participants saw the same scene, the more fixation points will be returned by our algorithm and the more meaningful the resulting fixation maps will be.

As we rely on image retrieval for this process, the captured objects in the video data should be recognizable from multiple viewpoints when using a feature descriptor. Generally, this restriction does not pose a problem and the proposed method is applicable to typical VR datasets.

3.2 Algorithm

In this section, we will describe our proposed algorithm for generation of fixation maps in dynamic free viewing virtual environments. Our pipeline utilizes a concept from feature-based image retrieval to project the gaze of other users from different viewpoints into the current view. Our approach can generally be divided into four sub-steps: (i) detection of SIFT keypoints and computation of image descriptors, (ii) generation of a vocabulary tree, (iii) gaze re-projection, and (iv) a fixation map calculation step.

Detecting SIFT keypoints: As the first step of the algorithm, we detect SIFT keypoints and compute corresponding descriptors of all frames (Lowe, 2004). Each descriptor contains histograms of the gradients in the local neighborhood of a keypoint, thus capturing the texture around the keypoint. The method is largely invariant to changes in scale, illumination, and rotation, making it a robust approach to re-detect image regions from different viewpoints. While the amount of keypoints and feature descriptors varies depending on the input frame, we obtain on average 2482 keypoints per image, each keypoint described by a 128 dimensional SIFT descriptor.

Computing the vocabulary tree: To speed up the lookup process of similar frames, we compute a vocabulary tree (Nister and Stewenius, 2006) for each individual environment. A vocabulary tree quantizes the extracted keypoints into visual words using a clustering algorithm, typically with k-means clustering. The quantization process is repeated in a hierarchical manner, constructing a tree structure. This structure allows a quick traversal and fast matching of SIFT descriptors. For the generation of the vocabulary tree we took the individual videos of all participants of two datasets, *DGaze* and *Saliency in VR* , used later in our evaluation, see Sec. 4.1 . Assuming that adjacent frames of the input only change slightly from frame to frame, we sub-sampled every 15th frame of the *DGaze* dataset and every 10th frame of the *Saliency in VR* dataset for the creation of the vocabulary tree to reduce memory and computational requirements. Moreover, due to the number of descriptors, we additionally utilize mini batch k-means (Sculley, 2010) as our hierarchical clustering algorithm in addition to the hierarchical k-means algorithm traditionally used in vocabulary trees. Mini batch k-means was initially introduced as a k-means clustering algorithm for massive datasets with regard to web scale applications, such as duplicate detector or grouping of data. In contrast to k-means, it clusters the input by randomly picking a batch that contains a subset of the input elements rather than the whole dataset. Then instead of re-assigning the cluster labels and re-calculating the cluster centers each iteration for the whole dataset, the algorithm computes the labels for each batch and then updates the cluster centers using a learning rate, interpolating between the cluster center and the new point. The algorithm has the advantage of reduced stochastic noise compared to online stochastic gradient descent, offering good convergence speed to a near optimal solution. However, due to its properties, we also utilize it to generate the vocabulary tree depending on the number of descriptors. Therefore, we either utilize mini batch k-means if the number of descriptors succeeded a specified threshold and use classical hierarchical k-means otherwise.

Gaze re-projection: The core of our algorithm, schematically shown in Figure 1 with another concrete example depicted in Figure 3 , is the re-projection of gaze points from similar frames onto a query frame using a projective transformation between the frames. This projection is required as the target frame is typically not aligned with the frames returned by the vocabulary tree. Also, as a gaze point can be treated like a pixel, it is possible to transfer gaze points of similar frames onto the input frame using a homographic projection matrix. A homography matrix is defined as a perspective projection that operates on homogeneous coordinates (Szeliski, 2010). Therefore, H can be used to project a point inside a source image plane p into a projected point p' of a target image plane:

$$p' = Hp. \tag{1}$$

As $p = (p_x, p_y, 1)^\intercal$ this calculation can further be simplified into:

$$\tilde{x}'_x = \frac{h_{00}p_x + h_{01}p_y + h_{02}}{h_{20}p_x + h_{21}p_y + h_{22}}, \quad \tilde{x}'_y = \frac{h_{10}p_x + h_{11}p_y + h_{12}}{h_{20}p_x + h_{21}p_y + h_{22}}. \tag{2}$$

To find frames similar to our target frame, we query the earlier constructed vocabulary tree. However, instead of using all descriptors of the target to find similar frames, we restrict the descriptors to a small

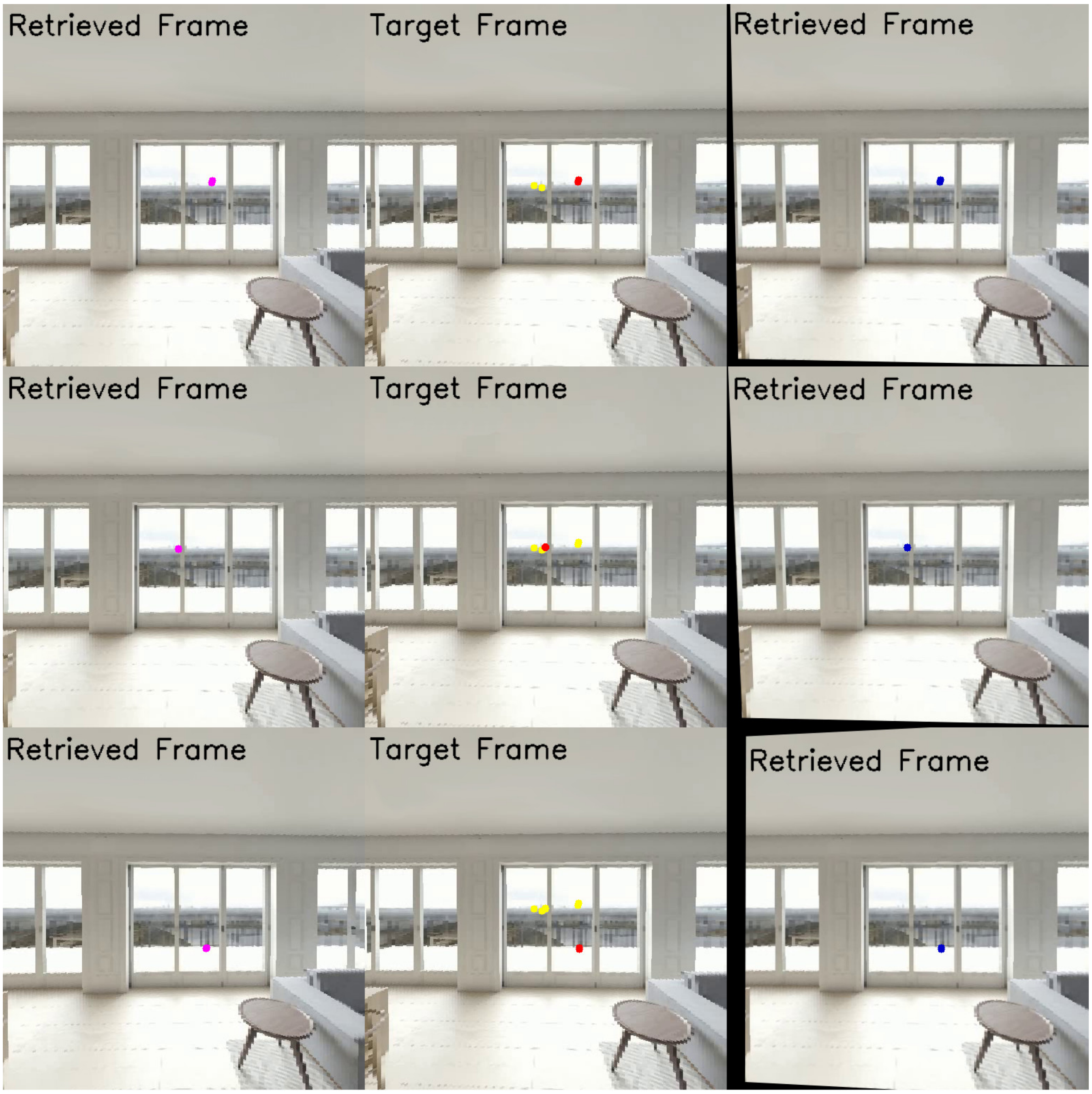

Figure 3. Gaze mapping from the retrieved frames on the left onto a single target frame in the center. The re-projection of the retrieved frame to match the target is shown on the right. The yellow dots show the gaze data that is already contained on the target frame, either because they belong to the target frame or through previous mapping steps. The magenta dots show the gaze retrieved with the retrieved frame. These blue dots correspond to their magenta counterparts, fixating the same position inside the image. These may, however, not have the same pixel location as the original magenta dots due to their projection. After their projection, we have shown the mapped gaze point as a red dot in the target frame, which is different from the retrieved frame. The shown re-projection is repeated multiple times to acquire the full fixation map.

128×128 subregion as shown in Figure 4 of our 512×512 target image, approximately corresponding to a 36° visual angle. As the center of the patch, we use the active gaze point of our target. This region, which refer from now as foveal-region, should cover at least the area that is covered by the fovea of the

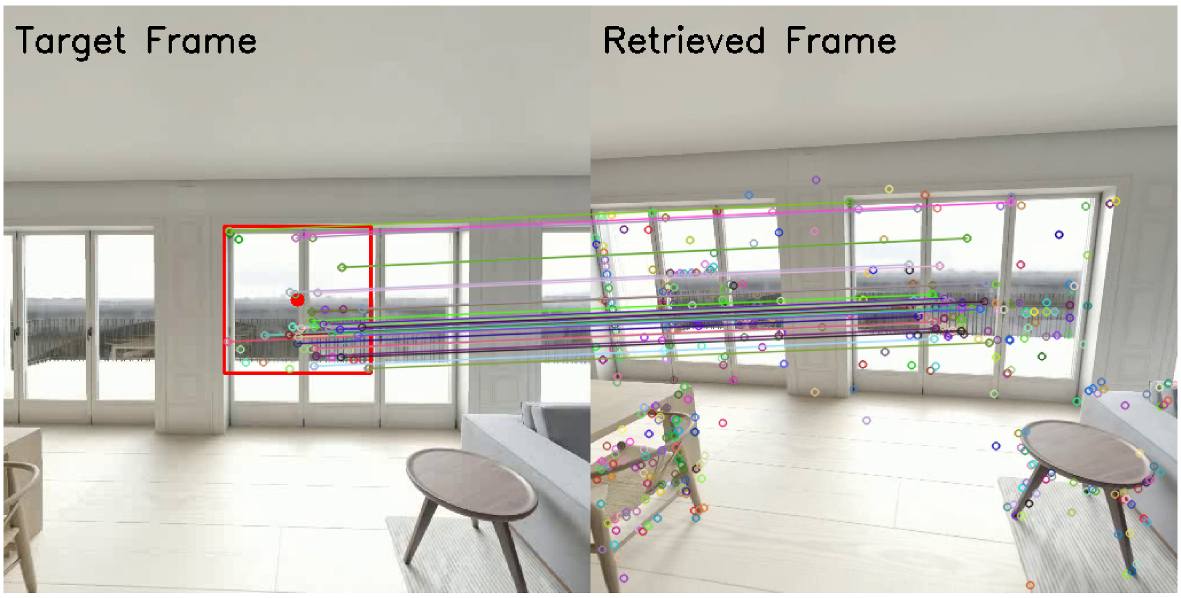

Figure 4. Example showing the matching process to find similar key-points between the retrieved and the query image. The small circles correspond to the positions of the image descriptors and the connected lines to the found matches. Note that the query frame uses a restricted area (red rectangle) around the current gaze position (red dot) that roughly covers the area visible in the fovea of the eye. The region is used to search for frames similar to the query frame.

eye. Querying the vocabulary tree using the extracted foveal-region then returns frames that are not only successors or predecessors of the query frame. For this reason, we chose an image patch size of 128×128 for the evaluated datasets, as otherwise the region might not contain enough descriptors for a reasonable mapping between the retrieved and the input frames. We would also like to note that a bigger foveal region might result in more accurate matches between the input and the retrieved frames. This is since the vocabulary tree returns frames that are most likely very close to the input, such as adjacent frames or frames that show the same content. Additionally, the increased amount of keypoints used to estimate the homography matrix will factor into a more accurate matching. However, this will also decrease the likelihood of mapping gaze points from other users into the target sequence. To estimate the homography matrix from the matching between the input and similar frames we utilize random sample consensus (RANSAC) (Fischler and Bolles, 1981) as some matches might not be correct, thus, resulting in noise.

Using the retrieved similar images, we match to most similar images with the query frame using the descriptors earlier, via the matching method proposed by Lowe (2004). For this, we utilize the two nearest neighbors of the descriptor and discard any found matching that are above a specified ratio. This results in a set of matches that map a part of the keypoints of the retrieved image K_r with a part of the target K_t keypoints. Figure 4 shows such an example of matched frames. It is also noteworthy that we now extract the foveal-patch for each similar image at the gaze position while utilizing all image descriptors of the query frame. This allows us to find the actively attended foveal-region inside the query even if the rest of the retrieved frame does not match the query frame.

Furthermore, to stabilize the found gaze over time, we do another gaze re-projection pass afterwards. For this pass, we only utilize gaze points of the currently processed video by explicitly querying the vocabulary tree to find similar frames in the query video. While this mostly results in frames adjacent to the query frame, it additionally retrieves frames that are similar but from other viewing angles, allowing to transfer additional information that was previously not found through the re-projection method.

Fixation map computation: In the last step, we compute the final eye fixation map equivalent to previously proposed methods for eye fixation map generation in static setups (Itti et al., 1998; Cornia et al., 2018), by applying a Gaussian filter to the previously computed eye fixation map with a sigma of 19 to generate the saliency map, as commonly used in other datasets Jiang et al. (2015) with the suggestion of Bylinskii et al. (2018).

To summarize, our algorithm can be expressed through the following steps:

- Extract features for vocabulary tree creation and image matching from all videos showing the scene.
- Build vocabulary tree using the descriptors of all recorded videos and insert each frame into the vocabulary tree.
- Compute the gaze re-projection for each frame of the target video by querying the vocabulary tree to find similar frames from videos of other users.
- Compute gaze re-projections across the initially generated eye fixations to transfer gaze points of adjacent and similar frames inside the target video.
- Compute Saliency map from the generated eye fixation maps.

4 EVALUATION

As we aim to provide eye fixation maps that indicate possible future fixation points for each frame in an input video to train gaze predictors, we would ideally evaluate against a set of likely fixation points for each target frame. Even though our evaluated datasets contain the raw gaze, we cannot directly utilize them for evaluation as it would require a mapping approach to first generate the ground truth data which we aim to provide. Such mapping approach then might also introduce a new error source when evaluating the approach. Therefore, under optimal conditions, it would require a hand mapped dataset of multiple videos. Since such data is not available, we measure how well our eye fixation maps model the gaze of the current user instead. We will, however, use metrics well established for comparing saliency predictors, as these provide well explainable results (Bylinskii et al., 2018). This also allows us to compare against other saliency predictors that could potentially be used to predict future gaze points.

For the evaluation itself, we utilize the *DGaze* (Hu et al., 2020) and the *Saliency in VR* (Sitzmann et al., 2018) datasets and evaluate by mapping the fixation point of each user given the current visual stimulus. We specifically choose the *DGaze* dataset to evaluate how well our algorithm maps fixation points in regard to the exploration of a single user, as all videos and gaze sequences in the *DGaze* dataset are unique. For the evaluation of our algorithm, we will generate saliency maps from the fixation maps and compare them with the actual fixation points. For the *DGaze* dataset we will compare only against the fixation points of a single user, whereas for the *Saliency in VR* dataset we evaluate against the fixation points of all users. Using saliency maps also enables us to utilize commonly used metrics to compare saliency maps, listed in Section 4.2. Due to the novelty of our contribution we cannot compare against a baseline algorithm, instead we compare our results against well established algorithms for visual saliency computation that predict the fixation distribution of an image. In short, we will compare against the following saliency predictors:

- Boolean Map Saliency (BMS) (Zhang and Sclaroff, 2013)
- Minimum Barrier Salient Object Detection (MB, MB+) (Zhang et al., 2015)
- VOCUS2 (Frintrop et al., 2015)
- Salient Attentive Model (SAM) (Cornia et al., 2018)

However, to avoid that our algorithm has an unfair advantage against the saliency predictors, as the algorithm has all gaze fixations available to it, we restrict the evaluation to use cross validation. Here, we only utilize gaze points of the database that do not belong to the currently processed video. Therefore, the algorithm only has the fixation points of other participants available to it. We, additionally, evaluate the algorithm by using past gaze points, mimicking a real-time application of our algorithm where a user is actively exploring the environment. In this case, fixations are computed given the currently rendered frame by comparing it against the internal database. Nonetheless, in both cases the assumption is that multiple participants will attend to similar or the same visual stimuli, resulting in similar fixation positions. This additionally measures the inter-observer congruency. In our case, the data points of all users except the i^{th} are used to estimate the saliency map of the i^{th} user. Here, we want to point out that this is the reverse of the inter-observer congruency measured by Sitzmann et al. (2018), as they measure how well a singular user i can model the fixations of all others.

4.1 Datasets

For the evaluation itself, we compare the generated saliency maps with the ground truth of two datasets. First, we evaluate against the *DGaze* dataset (Hu et al., 2020). This dataset captures individual free viewing of 43 participants that were asked to freely explore 2 out of 5 randomly assigned virtual environments without any restrictions on viewpoint, positioning or execution order of temporal events. The scenes shown to each participant also contain multiple dynamic objects in the form of animals that move randomly across the environment. At the start of a recording session, each participant were instructed to record at least 3 minutes of data without any further specifications on a task. To avoid any auditory disturbance, each participant was also provided with earplugs. In total, the dataset contains 1,789,082 gaze points and 1,046,467 frames with an average sequence length of 20,803 gaze points and 12,168 frames. The videos capturing the visual stimuli are stored with a resolution of 540×600 pixels. To capture the data, an HTC Vive in combination with a 7invensun eye-tracker was used. Further, all participants were provided with a HTC Vive controller, allowing them to teleport to any position inside the scene. To familiarize themselves with the experimental setup, everyone was given 3 minutes to learn the system provided controls. This specific approach to capture the data implies that participants rarely saw the exact same content that another participant has seen. When evaluating any algorithm on this dataset it is, therefore, only possible to directly compare the captured gaze points that were captured for the specific frame the participant has seen. While the amount of captured gaze points depend on the polling rate of the eye-tracker, in case of the *DGaze* dataset it results in one or two gaze points per captured frame.

In addition, we evaluate our system against the *Saliency in VR* dataset by Sitzmann et al. (2018). This dataset provides 22 static omnidirectional 360° images, containing in gaze sequences of 169 participants that were captured in VR. Here, Sitzmann et al. choose static omnidirectional images with a fixated head position to work around the exact same restrictions we aim to solve. However, this restriction of the viewpoint of every participant allows to directly evaluate the gaze of different users, as all gaze points can be mapped onto the omnidirectional image. We, therefore, extracted individual videos of every participant showing their exploration of the environment as seen through the HMD. It also enables the generation of saliency maps containing the gaze of all participants, solving the problem of providing only few ground truth points per frame. To acquire the data, an Occulus DK2 was used in combination with a pupil labs eye-tracker to capture the raw gaze points. Further, each participant was instructed to naturally explore the displayed environment for 30 seconds. Now, as the dataset contains the gaze and the head orientation of each participant, it is possible to simulate similar conditions to the *DGaze* dataset. Thus, we extracted a subset of 137 individual exploration videos of all users showing the visual stimuli along with the gaze data. This resulted in a total of 482,805 frames and gaze points. We additionally restricted the visual fixations to a center region to simulate center biased virtual reality vision similar to the *DGaze* dataset by restricting the gaze points to be inside a range of $[-35°, 35°]$ as Hu et al. reported that 98.7% of gaze points lie inside this area (Hu et al., 2020). Such a center bias in VR is also reflected by the findings of Celikcan et al. (2020) and (Foulsham et al., 2011). We, additionally, measure the behavior of all algorithms with different field of views applied by restricting the image region of the generated saliency maps.

4.2 Metrics

To evaluate the generated saliency maps, we selected multiple different metrics based on the recommendation by Bylinskii et al. (2018). For reproducibility, we base our implementation on metrics provided by Bylinskii et al. (2020). The metrics we especially focus on are:

- *Area under curve (AUC)* as proposed by Judd and Borji and further denoted as *AUC-Judd* (Riche et al., 2013) and *AUC-Borji* (Borji et al., 2013). These metrics work under the interpretation of a saliency map as a classifier that classifies pixels into two categories that classifies if a pixel is fixation or not. As both metrics use binary classification for each pixel, it is also possible to re-interpret the metrics as measuring a two-alternative forced choice task (Bylinskii et al., 2018). With these two categories it is possible to assign each pixel a label as either true positive, false positive, true negative or false negative. The main difference between both metrics is the calculation of the false positive rate, which is defined as the fraction of wrongly positive assigned pixels divided through the total number of pixels. Reason for the selection of the metric being the location independence of AUC-metrics, therefore, measuring the ability of a model to predict the correct salient spots and their in-variance to linear transformations such as contrast.
- *Normalized Scanpath Saliency (NSS)* measures the average normalized saliency at fixation locations (Bylinskii et al., 2018). Thus, the metric can be understood as a correspondence measure between a saliency map and the ground truth. This allows the measurement to be invariant against linear transformations, such as contrast offsets, however, it also makes the metric sensitive to false positives, relative differences in saliency and monotonic transformations.
- *Similarity (SIM)* measures the similarity between two probability distributions. This works under the assumption that a saliency map can be interpreted as a probability map. While it is not possible to directly observe the ground truth of the distribution, it is often approximated by blurring the fixation points (Bylinskii et al., 2018). However, using Gaussian blurring directly affects the similarity metric, as it results in the highest score if the used sigma for blurring matches the ground truth. Otherwise, if the sigma does not match, a drastic reduction in SIM score can be noticed Bylinskii et al. (2018). Nevertheless, the metric was chosen for its symmetrical behavior, thus allow an evaluation on partial matches.
- *Kullback-Leibler Divergence (KLD)* is similar to SIM a probability-based metrics as it measures the difference between two probability distributions with a lower score indicating a better approximation. Further, KLD was chosen for its penalization of sparse predictions as zeroes are heavily punished.
- *Pearson cross correlation (CC)* describes another distribution-based metric that measures the correlation between two dependent variables. Furthermore, CC treats, in contrast to SIM, true positives and false negatives symmetrically.

4.3 Results & Discussion

Table 1 and 2 show how well our gaze mapping approach is able to model the gaze of the current user, compared to computationally created saliency maps. First, we provided a qualitative overview of the evaluated methods on the *Saliency in VR* dataset in Fig. 5. We performed a quantitative evaluation on

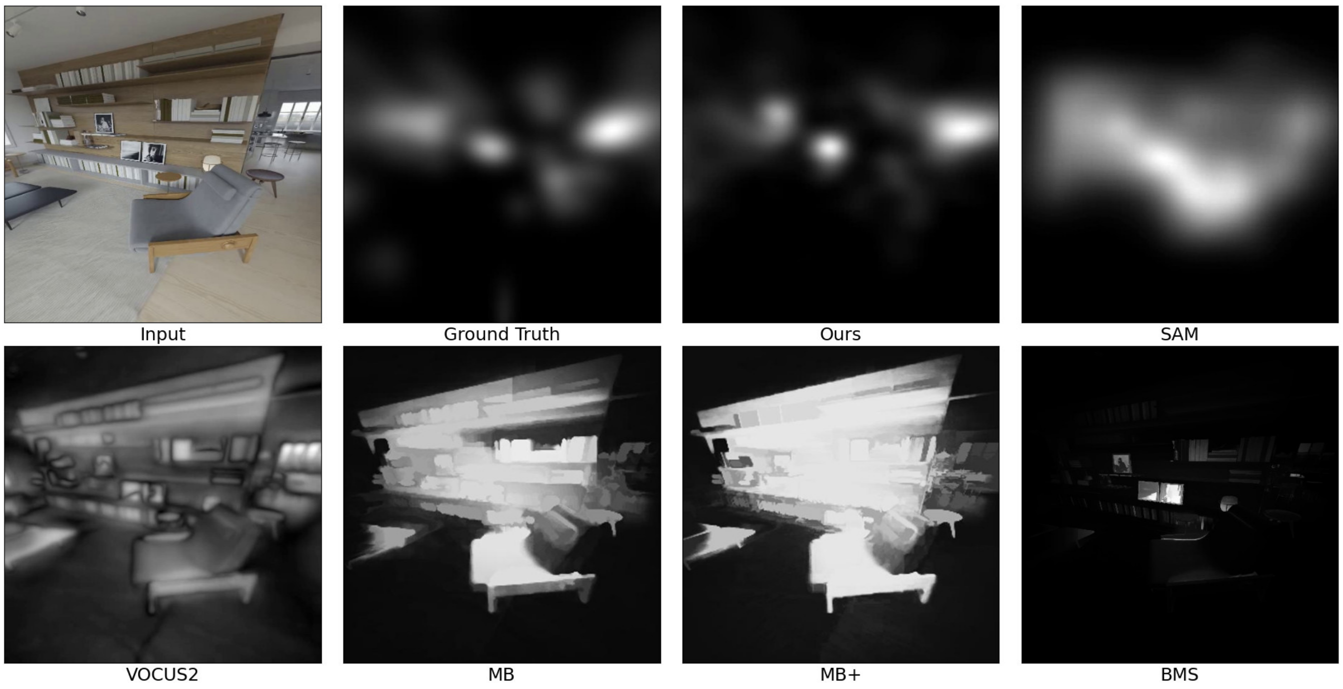

Figure 5. Qualitative example showing the output of the evaluated saliency predictors along with the ground truth and our approach on the *Saliency in VR* dataset. Our method uses the eye fixations from other users and maps them to the current view, without using the fixations from the current user, while the standard saliency predictors compute saliency based on the current frame.

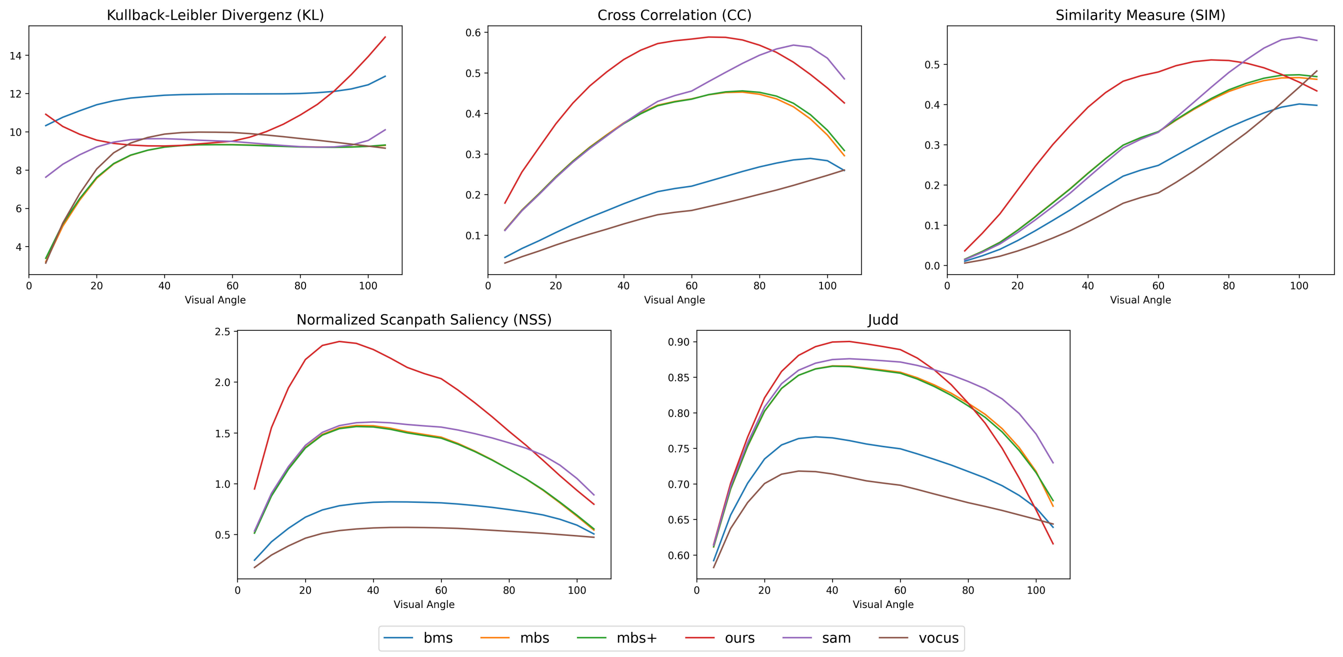

Figure 6. Graphs showing the results measured on the *Saliency in VR* dataset with different field of view angles to simulate different VR headsets and viewing conditions. The figure shows the accumulated measurements over our extracted subset of the dataset mentioned in Section 4.1. This subset contains 4 different omnidirectional images containing 137 videos totaling an amount of approximately 500k frames. Section 4.2 explains the show metrics in detail.

the *DGaze* and *Saliency in VR* datasets. In total, we evaluated over 1.5 million images on 223 videos. Table 1 shows the results measured using the *DGaze* dataset. We evaluated our algorithm on the *DGaze* dataset using two different modes of our method. First, we use cross-validation without previous temporal information of the input, referred to as *Ours* in Table 1. Additionally, we utilized temporal information about the input, referred as *Ours Temp* in Table 1. The later one simulates a real-time application of our algorithm, as this would provide a set of previous data captures from gaze points and images from other users. In contrast, the first one estimates the gaze captured only from other users given a video. Note that if we restricted our approach to only utilize past gaze data, it would still be required to map gaze from other frames, as those may not be aligned with the current one. In this case, utilizing the data from past gaze points also provides information to the system as other users might not have previously perceived part of the scene. It shows that our temporal algorithm almost always reliably outperforms all other methods on all evaluated metrics. This is to be expected, as the algorithm additionally uses past fixation points of the current user to generate the fixation maps. Therefore, the algorithm makes use of the gaze information and the similar visual stimuli. Thus, it is to be expected that the algorithm performs better, as no other algorithm makes use of the data and therefore cannot benefit from the information. However, when restricting our algorithm to use just the video sequences and gaze points from other users, our method almost always performs worse with regard to the KL and Judd metric and depending on the scene is sometimes outperformed by the MB, MB+ and SAM. This is also to be expected, since the most frames of the *DGaze* dataset are unique and do not show the same object or environment from the same direction. Hence, the algorithm only finds visual stimuli roughly similar to the input. We additionally found that our model sometimes results in a drop in performance if the visual stimulus is self-similar. Therefore, the algorithm will estimate the same fixation positions for different parts of the virtual environment.

Table 2 shows the results measured on the *Saliency in VR* dataset. Here, we found that our algorithm performs especially well on the all metrics and outperforms all other methods, except for the Kullback-Leibler divergence. Note that we do not use temporal information for the *Saliency in VR* dataset, as it only contains static omnidirectional images and therefore does not show content other users have not seen before. However, the performance of our algorithm can be explained by self-similarity of objects in the input, as it will retrieve images that show similar objects thus mapping fixation points from another object similar to the target. Further, the performance of our algorithm with regard to the KL metric can be explained by the sparse output of the fixation points, resulting in sparse predictions that are heavily punished by the KL metric (Bylinskii et al., 2018). Figure 6 also shows that depending on the restriction of the field of view, our algorithm exhibits better results if the field of view is smaller. This behavior can be explained by the calculation of the homography matrix, as our approach directly fits a homography matrix between the target frame and the retrieved frame. Now, if there is little overlap between those frames it might be that there are not enough corresponding image descriptors in both images for the computation of the projection matrix, resulting in an invalid or numerical unstable projections. This might also result in a center bias as more projections are discarded when there is little overlap between the frames, for example if both frames only overlap at their edges.

It is noteworthy that we can confirm the results of Celikcan et al. (2020), as we found that a good contender for approximating the saliency of a VR environment is the MB+ algorithm. However, we would like to stress once more that we do not aim to provide a saliency predictor, but rather a pipeline for the generation of ground truth fixation maps that can be utilized to train and evaluate such saliency predictors.

5 CONCLUSION & FUTURE WORK

In this paper, we proposed an algorithm to generate fixation maps from gaze points captured in free viewing virtual environments using a classical computer vision approach. Our algorithm was especially designed to compute fixation points given a visual stimulus in the form of an input video by collecting and projecting eye fixation points from other users showing similar content to the input using classical computer vision algorithms. To retrieve input stimuli similar to the input, we use an image retrieval approach based on SIFT features. This also allows us to project the retrieved images onto the target using a homography transformation for the gaze data to re-project the gaze into the current view. Since the algorithm can compute saliency maps from the projected fixation maps, it can also be utilized to generate ground truth fixation and saliency maps to train ML saliency models for virtual environments. We also showed that our algorithm can outperform recent state-of-the-art methods on several metrics on the *Saliency in VR* dataset. It is also worth noting that our algorithm will generate saliency maps similar to the input, if similar patterns or self-similarities are shown inside the input, as it will retrieve all frames containing similar visual stimuli even if the shown object is not the same.

However, we would also like to address some future improvements, as we aim to utilize our method to train saliency predictors in current gaze prediction pipelines using the generated fixation maps as ground truth to train these models. First, we currently do not utilize the temporal information of adjacent video frames for the calculation of the homographic projection which we will address in future work.

Another improvement would be the estimation of camera properties that are necessary for image distortion, as our algorithm currently does not distort the input. This might result in slight off gaze points calculated through the mapping and therefore do not mirror the actual gaze position inside the target frame. Further, extending our algorithm by using a 3-dimensional representation of the virtual space, either through Structure from Motion or by providing a model of the scene, would allow us to accurately map the gaze into 3D space. This also reduces the computational requirements as it only requires to project gaze points that are visible to the user at run-time. At last, we also do not explicitly deal with dynamic objects, such as animations or changes in appearance; hence, one could apply image segmentation to extract dynamic objects and process them separately. Furthermore, our results also show that scenes which contain dynamic animated objects will result in wrongly assigned fixation positions. This is especially true if the animation is highly salient in contrast to the default behavior, for example when gesturing something to the user. In these cases, our algorithm might fail because different descriptors describe the object. An option to solve these might be the extraction of dynamic objects to generate the gaze mapping of those independent on the scene.

	Algorithm	KL$_\downarrow$	CC$_\uparrow$	SIM$_\uparrow$	NSS$_\uparrow$	Judd$_\uparrow$
Scene 1 Horses in desert	Ours	12.41	0.19	0.14	2.42	0.92
	Ours Temp	**10.82**	**0.29**	**0.19**	**3.67**	**0.96**
	BMS	16.06	0.12	0.07	1.56	0.83
	MB	11.16	0.23	0.08	2.76	**0.96**
	MB+	11.50	0.16	0.07	1.85	0.93
	VOCUS2	12.06	0.08	0.04	0.88	0.87
	SAM	10.87	0.27	0.14	3.26	**0.96**
Scene 2 Deers in forest	Ours	12.41	0.19	0.14	2.42	0.92
	Ours Temp	**11.42**	**0.25**	**0.17**	**3.05**	**0.95**
	BMS	16.85	0.02	0.03	0.21	0.75
	MB	11.60	0.14	0.06	1.60	0.93
	MB+	11.62	0.14	0.06	1.61	0.92
	VOCUS2	12.49	0.01	0.03	0.23	0.79
	SAM	11.40	0.20	0.09	2.34	0.94
Scene 3 Chickens in warehouse	Ours	13.07	0.12	0.08	1.51	0.90
	Ours Temp	11.19	**0.26**	**0.18**	**3.28**	0.95
	BMS	15.50	0.07	0.05	0.90	0.81
	MB	11.26	0.20	0.08	2.42	**0.96**
	MB+	11.19	0.20	0.08	2.43	**0.96**
	VOCUS2	12.01	0.11	0.04	1.30	0.88
	SAM	**10.99**	0.23	0.12	2.84	**0.96**
Scene 4 Cats in bar	Ours	11.51	0.15	0.08	1.82	0.92
	Ours Temp	**10.82**	**0.25**	**0.14**	**3.09**	**0.96**
	BMS	13.53	0.06	0.04	0.67	0.81
	MB	11.70	0.13	0.06	1.52	0.91
	MB+	11.72	0.13	0.06	1.47	0.91
	VOCUS2	12.46	0.03	0.03	0.31	0.77
	SAM	11.55	0.19	0.09	2.22	0.93
Scene 5 Sheeps on meadow	Ours	11.41	0.17	0.10	2.04	0.94
	Ours Temp	**11.07**	**0.24**	**0.16**	**2.96**	**0.95**
	BMS	17.85	0.03	0.03	0.34	0.77
	MB	11.48	0.17	0.07	2.08	0.93
	MB+	11.46	0.17	0.07	2.02	0.93
	VOCUS2	12.04	0.08	0.04	1.02	0.87
	SAM	11.42	0.23	0.10	2.79	**0.95**
Total All scenes combined	Ours	12.31	0.15	0.09	1.77	0.91
	Ours Temp	**11.06**	**0.26**	**0.17**	**3.21**	**0.96**
	BMS	15.96	0.06	0.04	0.74	0.79
	MB	11.44	0.17	0.07	2.08	0.94
	MB+	11.50	0.16	0.07	1.87	0.93
	VOCUS2	12.21	0.06	0.03	0.75	0.84
	SAM	11.24	0.22	0.11	2.69	0.95

Table 1. Prediction capability of the gaze of the current user on the *DGaze* dataset (Hu et al., 2020) with our gaze mapping approach, either by including the previous gaze points of the current user (Ours Temp), or by ignoring those and focusing on gaze points from other users only (Ours).

	Algorithm	KL$_{\downarrow}$	CC$_{\uparrow}$	SIM$_{\uparrow}$	NSS$_{\uparrow}$	Judd$_{\uparrow}$
Cubemap 0000 Appartment Complex (inside)	Ours	**8.82**	**0.72**	**0.60**	**2.73**	**0.93**
	BMS	11.81	0.26	0.27	1.03	0.77
	MB	9.18	0.43	0.32	1.62	0.88
	MB+	9.11	0.44	0.33	1.66	0.88
	VOCUS2	9.75	0.21	0.18	0.82	0.75
	SAM	9.06	0.49	0.34	1.85	0.90
Cubemap 0001 Appartment Complex (outside)	Ours	9.93	**0.57**	**0.49**	**1.71**	**0.88**
	BMS	12.27	0.23	0.25	0.85	0.75
	MB	8.95	0.51	0.36	1.63	0.85
	MB+	8.95	0.50	0.37	1.59	0.85
	VOCUS2	**9.57**	0.24	0.20	0.74	0.72
	SAM	9.59	0.46	0.33	1.40	0.85
Cubemap 0002 Futuristic Shore City (inside)	Ours	9.90	**0.50**	**0.40**	**1.92**	**0.89**
	BMS	11.27	0.22	0.24	0.85	0.77
	MB	**9.73**	0.46	0.33	1.55	0.88
	MB+	9.78	0.43	0.32	1.47	0.87
	VOCUS2	10.47	0.11	0.16	0.48	0.70
	SAM	9.85	0.43	0.31	1.53	0.88
Cubemap 0003 Library (inside)	Ours	**9.35**	**0.44**	**0.33**	**1.56**	**0.86**
	BMS	12.81	0.10	0.15	0.34	0.70
	MB	9.53	0.31	0.24	0.98	0.82
	MB+	9.55	0.31	0.24	1.00	0.82
	VOCUS2	10.20	0.03	0.12	0.04	0.59
	SAM	9.86	0.37	0.25	1.31	0.84
Total All scenes combined	Ours	9.43	**0.58**	**0.47**	**2.08**	**0.89**
	BMS	11.97	0.21	0.24	0.82	0.75
	MB	9.33	0.43	0.32	1.48	0.86
	MB+	**9.32**	0.43	0.32	1.47	0.86
	VOCUS2	9.98	0.16	0.17	0.57	0.70
	SAM	9.52	0.44	0.31	1.57	0.87

Table 2. Prediction capability of the gaze of the current user on the *Saliency in VR* dataset (Sitzmann et al., 2018) with our gaze mapping approach (Ours).

REFERENCES

Albert, R., Patney, A., Luebke, D., and Kim, J. (2017). Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)* 14, 1–13

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding* 110, 346–359

Benjamins, J. S., Hessels, R. S., and Hooge, I. T. (2018). Gazecode: Open-source software for manual mapping of mobile eye-tracking data. In *Proceedings of the 2018 ACM symposium on eye tracking research & applications*

Borji, A., Tavakoli, H. R., Sihite, D. N., and Itti, L. (2013). Analysis of scores, datasets, and models in visual saliency prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*

[Dataset] Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., et al. (2020). Mit saliency benchmark. http://saliency.mit.edu/

Bylinskii, Z., Judd, T., Oliva, A., Torralba, A., and Durand, F. (2018). What do different evaluation metrics tell us about saliency models? *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41

Celikcan, U., Askin, M. B., Albayrak, D., and Capin, T. K. (2020). Deep into visual saliency for immersive vr environments rendered in real-time. *Computers & Graphics* 88, 70 – 82

Che, Z., Borji, A., Zhai, G., Min, X., Guo, G., and Le Callet, P. (2019). How is gaze influenced by image transformations? dataset and model. *IEEE Transactions on Image Processing* 29, 2287–2300

Cornia, M., Baraldi, L., Serra, G., and Cucchiara, R. (2018). Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing* 27

De Tommaso, D. and Wykowska, A. (2019). Tobiiglassespysuite: An open-source suite for using the tobii pro glasses 2 in eye-tracking studies. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. 1–5

Droste, R., Jiao, J., and Noble, J. A. (2020). Unified Image and Video Saliency Modeling. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24

Foulsham, T., Walker, E., and Kingstone, A. (2011). The where, what and when of gaze allocation in the lab and the natural environment. *Vision Research* 51

Frintrop, S., Rome, E., and Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. *ACM Trans. Appl. Percept.* 7

Frintrop, S., Werner, T., and Martin Garcia, G. (2015). Traditional saliency reloaded: A good old model in new shape. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 82–90

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., and Van de Weijer, J. (2011). *Eye Tracking : A Comprehensive Guide to Methods and Measures* (OUP Oxford)

Hosny, Y. S. S., Salem, M. A.-M., and Wahby, A. (2020). Performance optimization for standalone virtual reality headsets. In *2020 IEEE Graphics and Multimedia (GAME)* (IEEE), 13–18

Hu, Z., Bulling, A., Li, S., and Wang, G. (2021). Fixationnet: Forecasting eye fixations in task-oriented virtual environments. *IEEE Transactions on Visualization and Computer Graphics* 27, 2681–2690

Hu, Z., Li, S., Zhang, C., Yi, K., Wang, G., and Manocha, D. (2020). Dgaze: Cnn-based gaze prediction in dynamic scenes. *IEEE Transactions on Visualization and Computer Graphics* 26

Huang, Y., Cai, M., Li, Z., and Sato, Y. (2018). Predicting gaze in egocentric video by learning task-dependent attention transition. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 754–769

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence* 20

Jiang, M., Huang, S., Duan, J., and Zhao, Q. (2015). Salicon: Saliency in context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Kanter, D. (2015). Graphics processing requirements for enabling immersive vr. *AMD White Paper*

Koch, C. and Ullman, S. (1987). Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence* (Springer)

Kraus, M., Kilian, T., and Fuchs, J. (2019). Real-time gaze mapping in virtual environments. In *EUROVIS 2019: 21st EG/VGTC Conference on Visualization*

Kummerer, M., Wallis, T. S., Gatys, L. A., and Bethge, M. (2017). Understanding low-and high-level contributions to fixation prediction. In *Proceedings of the IEEE International Conference on Computer Vision*. 4789–4798

Kurzhals, K., Hlawatsch, M., Seeger, C., and Weiskopf, D. (2016). Visual analytics for mobile eye tracking. *IEEE Transactions on Visualization and Computer Graphics* 23

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 91–110

MacInnes, J. J., Iqbal, S., Pearson, J., and Johnson, E. N. (2018a). Mobile gaze mapping: A python package for mapping mobile gaze data to a fixed target stimulus. *Journal of Open Source Software* 3

MacInnes, J. J., Iqbal, S., Pearson, J., and Johnson, E. N. (2018b). Wearable eye-tracking for research: Automated dynamic gaze mapping and accuracy/precision comparisons across devices. doi:doi.org/10.1101/299925. Preprint

Mathe, S. and Sminchisescu, C. (2012). Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *European Conference on Computer Vision (ECCV)*. 842–856

Nakashima, R., Fang, Y., Hatori, Y., Hiratani, A., Matsumiya, K., Kuriki, I., et al. (2015). Saliency-based gaze prediction based on head direction. *Vision Research* 117, 59–66

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (IEEE), vol. 2

Pashler, H. E. (1999). *The psychology of attention* (MIT press)

Patney, A., Salvi, M., Kim, J., Kaplanyan, A., Wyman, C., Benty, N., et al. (2016). Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)* 35

Pfeiffer, T., Renner, P., and Pfeiffer-Leßmann, N. (2016). Eyesee3d 2.0: Model-based real-time analysis of mobile eye-tracking in static and dynamic three-dimensional scenes. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. 189–196

Riche, N., Duvinage, M., Mancas, M., Gosselin, B., and Dutoit, T. (2013). Saliency and human fixations: State-of-the-art and study of comparison metrics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision* (IEEE), 2564–2571

Sculley, D. (2010). Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*. 1177–1178

Sitzmann, V., Serrano, A., Pavel, A., Agrawala, M., Gutierrez, D., Masia, B., et al. (2018). Saliency in vr: How do people explore virtual environments? *IEEE transactions on visualization and computer graphics* 24, 1633–1642

Stein, N., Niehorster, D. C., Watson, T., Steinicke, F., Rifai, K., Wahl, S., et al. (2021). A comparison of eye tracking latencies among several commercial head-mounted displays. *i-Perception* 12

Szeliski, R. (2010). *Computer vision: algorithms and applications* (Springer Science & Business Media)

Treisman, A. M. and Gelade, G. (1980). A feature-integration theory of attention. *Cognitive psychology* 12

Treisman, A. M. and Kanwisher, N. G. (1998). Perceiving visually presented objets: recognition, awareness, and modularity. *Current opinion in neurobiology* 8

Uriza, E., Gómez-Fernández, F., and Rais, M. (2018). Efficient large-scale image search with a vocabulary tree. *Image Processing On Line* 8

Zhang, J. and Sclaroff, S. (2013). Saliency detection: A boolean map approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*

Zhang, J., Sclaroff, S., Lin, Z., Shen, X., Price, B., and Mech, R. (2015). Minimum barrier salient object detection at 80 fps. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*