DOI: 10.1111/cgf.14176

COMPUTER GRAPHICS *forum*
Volume 00 (2020), number 00 pp. 1–14

Time-Warped Foveated Rendering for Virtual Reality Headsets

Linus Franke,[1] Laura Fink,[1] Jana Martschinke,[1] Kai Selgrad[2] and Marc Stamminger[1]

[1]Computer Graphics Group, University of Erlangen-Nuremberg, Germany
{linus.franke, laura.fink, jana.martschinke, marc.stamminger}@fau.de
[2]OTH Regensburg, Germany
kai.selgrad@oth-regensburg.de

Abstract

Rendering in real time for virtual reality headsets with high user immersion is challenging due to strict framerate constraints as well as due to a low tolerance for artefacts. Eye tracking-based foveated rendering presents an opportunity to strongly increase performance without loss of perceived visual quality. To this end, we propose a novel foveated rendering method for virtual reality headsets with integrated eye tracking hardware. Our method comprises recycling pixels in the periphery by spatio-temporally reprojecting them from previous frames. Artefacts and disocclusions caused by this reprojection are detected and re-evaluated according to a confidence value that is determined by a newly introduced formalized perception-based metric, referred to as confidence function. The foveal region, as well as areas with low confidence values, are redrawn efficiently, as the confidence value allows for the delicate regulation of hierarchical geometry and pixel culling. Hence, the average primitive processing and shading costs are lowered dramatically. Evaluated against regular rendering as well as established foveated rendering methods, our approach shows increased performance in both cases. Furthermore, our method is not restricted to static scenes and provides an acceleration structure for post-processing passes.

Keywords: perceptually based rendering, real-time rendering, rendering, immersive VR, virtual environments

ACM CCS: • Computing methodologies → Perception; Rasterization; Virtual reality

1. Introduction

Rendering photo-realistic images for virtual reality headsets or *head-mounted displays (HMDs)* is a demanding task due to the high computational cost and strict frame-rate constraints [Vla16]. Exploiting weaknesses in the *human visual system (HVS)* is an established way to loosen constraints and to reduce computation times in rendering [GFD*12, MDZV18, PSK*16, WRK*16]. One of these weaknesses is a less accurate perception in the periphery of the gaze direction; methods targeting this are commonly referred to as *foveated rendering* techniques. An extensive summary about the human visual system and how its weaknesses can be exploited is given by Weier *et al*. [WSR*17]. Generally for foveated rendering, the physiology of the *retina* (the eye's sensory system, located at the opposite side of the lens) is of main interest: It is comparable to a camera as light is guided by the lens onto the retina, where two kinds of receptors, *cones* and *rods*, produce different stimuli.

Cones are responsible for processing all visible wavelengths of light [GB16] and are mostly located directly at the *fovea*, a small (less than 10°) area where light from fixated areas is focused to by the lens. Cones are receptive to colours and are distributed in a dense pattern allowing sharp and detailed vision. In contrast to that, rods are colour-indifferent but motion-sensitive sensors and provide general information such as brightness or movement [GB16]. Across the complete retina, their number is about 20 times higher than that of cones, but their distribution is shifted towards the periphery such that no rods are present in the fovea and the highest concentration is at about 17° of the main view vector (gaze direction) [CSKH90]. With that, the human field of view can be abstracted by a model with a foveal region that exhibits sharp, detailed vision and a peripheral region which is primarily sensitive to brightness and motion.

Foveated rendering methods exploit this difference of perception in the visual field to accelerate rendering. A common problem in foveated approaches is geometric aliasing in the peripheral vision, which occurs as flickering and is thus easily noticeable due to the high density of rods in this area [WSR*17, GFD*12]. In our approach, we synthesize the peripheral region in reduced quality using *reprojection* (also called *warping*) from previous frames (similar to

Figure 1: *Our time-warped foveated rendering method produces visually plausible results (left, twice at 1280 × 1440 pixels in 4.97ms on a RTX 2070) as compared to regular rendering (right, 7.81ms). While the fovea (which is tracked, indicated here in white) is rendered regularly, the periphery is reprojected, and thus less accurate, however without being noticeable to the observer's eye and in a temporally stable manner. Consequently, an image perceptually on par is computed with a speed-up of almost 1.6×.*

a render cache [WDP99]), while rendering the foveal region in full detail every frame. For all peripheral pixels, we evaluate a confidence measure (an eye receptor-based formalization of reprojection quality) and enforce redrawing of peripheral pixels with low confidence. This way, reprojected pixels can survive for many frames, given that our confidence in them remains stable, resembling filtering for adaptive frame rendering [DWWL05]. The confidence measure also decides whether unavoidable holes in the reprojection can be interpolated or are to be filled with new, redrawn samples. Our method also handles dynamic objects and their reprojection through adding them into the confidence function.

Our approach is similar in spirit to Weier *et al.*'s [WRK*16], who integrated foveated reprojection into a real-time *ray-tracing* pipeline, where redrawing single pixels can be done relatively efficiently. The contribution of our paper is to show how to integrate temporal foveation into a *rasterization* pipeline. For that rendering paradigm, several additional considerations are required to avoid expensive redrawing operations without visible impact on image quality, thus the contribution of our paper are the following:

- A high-performance rasterizing foveated temporal rendering method, achieving stable renderings with reduced, but sufficient quality in the periphery, see Figure 1.
- The *confidence function*, a novel perception-based metric to evaluate the reprojection quality.
- Applications of the confidence function, allowing efficient hierarchical foveation culling and support for a wider range of scenarios with dynamic objects and ambient occlusion.

We also calibrate and validate our technique with two user studies, each with over 20 participants.

Following an overview of related work, there is a detailed description of our time-warped foveated rendering method: After providing details on our forward reprojection scheme (Section 4), we present our hole filling method and define our confidence function (Section 5). Furthermore we describe how its result is used to efficiently compute the missing image parts and illustrate how our rendered and reprojected image is composed and made stable for display in the HMD (Section 6), Additionally, we expand on how our technique supports dynamic objects and accelerates post-processing (Section 7). We conclude with an extensive evaluation (Section 8), presenting our user studies and performance measures and discussing limitations posed by our method. To outline one important limitation first, as we are reprojecting the colour of pixels to the periphery we need to assume some kind of consistency between images, thus we limit ourselves to non-moving lights in our method, as will be discussed in the evaluation (Section 8).

2. Related Work

Image Warping. The application of geometric transformations to an input image is a well established area of research. McMillan presents a foundation for image warping techniques in his dissertation [McM97], which acts as a baseline for techniques presented here. Generally, two different variants are common: backward and forward warping. The former, backward warping, denotes methods that gather information from previous frames based from fixed destination positions, while the latter, forward warping, refers to methods that scatter information from previous frames starting from fixed source positions. Warping techniques are used in many different scenarios: Nehab *et al.* [NSL*07] present a technique for caching shading computations by backward reprojection, reducing overall computation cost. Lee *et al.* [LKE18] use backward reprojection with fixed-point iteration [BMS*12] to predict a new frame's depth buffer, e.g. for occlusion culling. Smit *et al.* [SvLBF09] present an evaluation of an intermediate frame warping technique in a remote-rendering setup. Asynchonous time warp [VW16] as used by the Oculus HMD's driver warps the image between rendering and displaying to adjust to latest head tracking information. Recent advances in temporal upsampling techniques using image warping (called checkerboard rendering [dCI17]) have shown great success in reducing overall rendering costs.

Warping is also used for creating new images in full. Yang *et al.* [YTS*11] use bidirectional reprojection, i.e. forward and backward reprojection, to reconstruct an intermediate frame in-between two frames at the cost of a slight delay. Didek *et al.* [DRE*10] use forward reprojection for new frames and present an acceleration technique for this reprojection variant. They

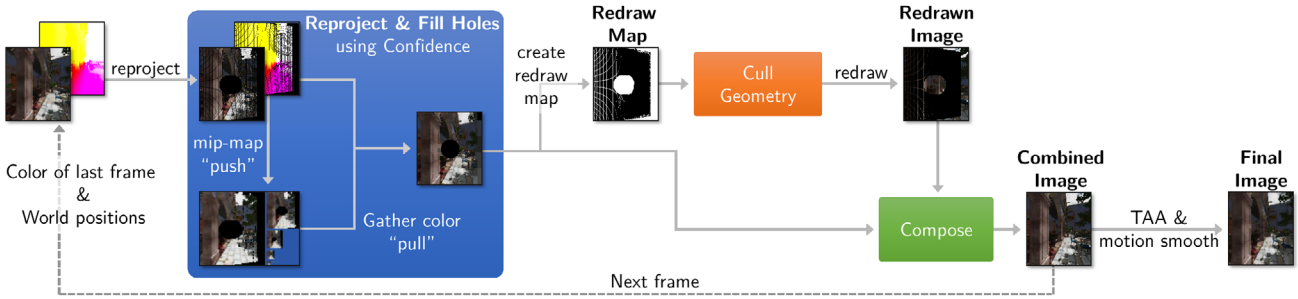

Figure 2: *Our time-warped foveated rendering pipeline: Last frame's colour and world position images are reprojected into this frame and hole-filled (blue). After that, a redraw map is computed based on our confidence in the reprojection. With that, geometry is culled (orange, objects fully in black areas in the map are culled) and missing areas (white in the redraw map) are rendered. Following, reprojected and redrawn images are composed (green), which results in the colour input for next frame. A final TAA and motion smoothing pass is applied to stabilize the image before sending it to the HMD. Note that the redraw-pass includes G-buffer rendering, shading and post-processing.*

group pixels with an adaptive grid, based on depth disparity, thus larger areas will be warped if their depth is similar. Schollmeyer *et al.* [SSB*17] improve on this concept with a tighter grid and support for transparency with an A-buffer [Eng14] based data structure, where transparent, rasterized fragments are stored and subsequently ray-traced from the new view. Without that, transparent fragments are difficult to warp as they are usually accumulated (alpha-blended) and have no distinct depths.

Foveated Rendering. In recent years, several foveated rendering approaches were introduced. Guenter *et al.* [GFD*12] generate three images for each frame, with progressively lower resolution for larger eccentricities, and combine them with bilinear upsampling, thus creating an image with full resolution in the fovea and low resolution in the periphery. With that, they reduce shading rates by up to 50%, but need to employ strong anti-aliasing methods to combat flickering, which is especially noticeable in the periphery. Patney *et al.* [PSK*16] circumvent this problem by decoupling visibility detection (which is always done in full resolution) from shading in their variable-rate shading-based foveated rendering method which they combine with temporal anti-aliasing [Kar14]. Inspired by coarse pixel shading [VST*14], the periphery is shaded at lower rates (such as one shading computation for 4×4 pixels) thus reducing shading rates by up to 75%. With both approaches, special care needs to be taken with artefacts caused by undersampling (such as shadow map aliasing) as lower resolution sampling rates in the periphery can contribute to the undersampling becoming more noticeable.

Adaptive sampling based on gaze is also used with foveated ray-tracing techniques. The general idea is to use more samples for foveal regions and minimal samples for the periphery to cut down the potentially high cost of ray tracing [FRS19, SGEM16, WRK*16]. Stengel *et al.* [SGEM16] create a saliency map and decide sampling rates based on it. This leads to more samples in visually important areas, including the fovea. Friston *et al.* [FRS19] exploit foveated rendering in a low-latency renderer for rolling displays in HMDs. Weier *et al.* [WRK*16] use a reprojection scheme to accelerate foveated ray tracing. While tracing the fovea in full, they use a lower-resolution G-buffer and reproject coarse pixel blocks to peripheral regions. Areas in the reprojection with strong depth differences are deemed problematic and error-prone and are thus resampled. For light field displays, where real-time performance is even more challenging, Sun *et al.* [SHK*17] investigated foveation in 3D perception, reducing the number of required rays by up to 84%. Tursun *et al.* [TAKW*19] propose a contrast-aware foveation concept for use in ray tracing and rasterization based methods, estimating required sample numbers. Meng *et al.* [MDZV18] present a different, analytical approach, where they use log-polar mappings to parameterize foveated rendering based on the receptor distribution on the retina and thus reducing cost of the rendering step.

Further supporting research aims to better understand influences on foveated rendering. To reduce perception of artefacts, Weier *et al.* [WRHS18] propose adding depth of field to hide inaccuracies and noise, which showed promising results in their user study. Swafford *et al.* [SIGK*16] introduce a foveated perceptual image metric, based on HDR-VDP2 [MKRH11], and parameterize it based on a desktop user study. Saliency in VR is explored by Sitzmann *et al.* [SSP*18], presenting that users tend to explore VR scenes very differently from each other, thus indicating that foveation without eye-tracking is challenging to predict accurately. Hoffmann *et al.* [HMT18] study the peripheral acuity in participants, concluding that lower resolution images shown in the periphery were not noticeable for up to 40 ms. This presents a possibility for other perception-based performance improvements for rendering, as e.g. exploited by Denes *et al.* [DMAM19] by interleaving frames rendered at full resolution with lower resolution ones. Albert *et al.* [APLK17] explore latency requirements of eye-tracking hardware. Their research suggests that a maximum latency of 50–70 ms from eye movement to displaying the image is tolerable to avoid noticeability, especially since the HVS omits information during fast eye movements (such as saccades).

3. Algorithm Overview

The general idea of our algorithm is to leverage information from previous frames for peripheral regions. The naive approach of reprojecting and only redrawing the fovea has problems though, as reprojection inherently produces artefacts. One class of artefacts are

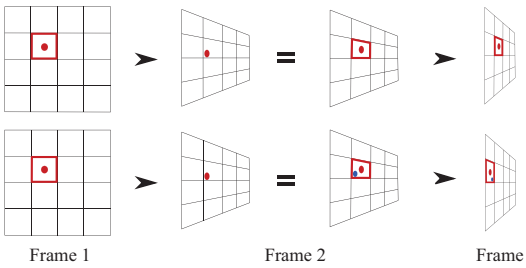

Figure 3: *Reprojection based on screen space depth (top row): Loss of exact subpixel position during rasterization (frame 2) leads to wrong reprojection in following frames (frame 3). Keeping fragment world space positions (bottom row, blue dot) allows correct rasterization in frame 3. This inaccuracy is avoided in the reprojection and exploited in our motion smooth pass to identify fragments likely flipping between two pixel footprints.*

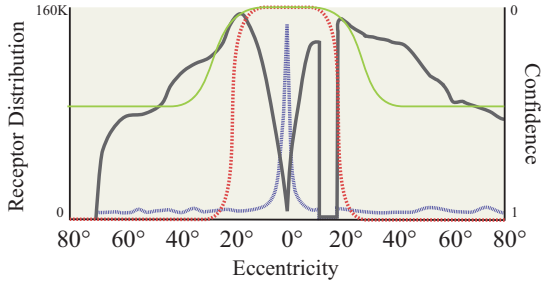

Figure 4: *Distribution of receptors (per square millimeter) and confidence: Low brightness areas target cones (blue), thus confidence (red) is closely matched to them. For high brightness areas, the confidence (green) additionally relies on rod distribution. Note that the confidence axis is inverted (confidence is zero at 0° eccentricity) for demonstration. Receptor distribution adapted from Weier* et al. *[WSR*17] and Goldstein [GB16].*

so-called *disocclusions*, i.e. areas hidden in previous frames that become visible in the new frame. A second class of artefacts is caused by mismatches in exact pixel locations for the reprojected data, i.e. small holes or inaccuracies in warping with coarse grid reprojection [WRK*16, SSB*17, DRE*10]. While some of the arising artefacts can be handled with cheap hole filling schemes, others must be solved by expensive redrawing. When targeting rendering for HMDs, deciding when to redraw or to fill holes is crucial, both in terms of performance, but even more in terms of image quality as artefacts are easier to notice in HMDs due to the larger perceived pixel sizes. For Weier *et al.*'s foveated ray tracing [WRK*16], this decision is easier, as resampling comes at a lower cost with ray tracing. In our rasterizing approach though, we need to avoid expensive pixel-perfect resampling, without compromising on perceived image quality. Therefore, we introduce a *general confidence function*, which is used to compute the confidence in the unnoticeability of the reprojection to formalize decision making.

Furthermore with our method, we strive to avoid these artefacts while still reprojecting as much information as possible from previous frames. Figure 2 displays our full pipeline. We start out with colour and position data from the previous frame, reproject and fill holes in the resulting image. Based on that we estimate how confident we are that the reprojection quality suffices, resulting in a set of screen space areas that need redrawing (the *redraw map*). We then render the thusly selected areas to obtain up-to-date pixel values, while further exploiting the redraw map to cull geometry (and subsequently fragments) not required for the indicated areas while rendering. Finally, the rendered and the reprojected pixel values are composed and undergo temporal anti-aliasing (TAA) [Kar14] and motion smoothing to provide a stable input to the HMD.

4. Reprojection

After retrieving the HMD's tracking information we use a uniform grid with a resolution equal to that of the input image for warping. Each cell of this grid is represented by a pixel-sized point sprite, which can be moved individually. This can be thought of as using the last frame's rasterized version of the scene for a second rasterization, with each point sprite containing colour (tonemapped 8-bits RGBA) and world positions (32-bits XYZW). In contrast to Weier *et al.* [WRK*16] using rasterized depth values for reprojection, we keep exact world pixel positions to ensure that our reprojection samples do not become inaccurate over time which otherwise can happen due to loss of sub-pixel positions (see Figure 3).

In general, small head movements and noise in the sensor data cause considerable motion between two images, thus two or more sprites frequently get warped to the same pixel footprint. This collision is resolved via hardware depth testing, thus discarding occluded pixels. When doing so, the resulting image likely has holes of varying sizes, which either need to be filled or redrawn (see next section). It is important to note that we are not limiting our method to always have a fresh, fully rendered image as input (as, e.g., with Yang *et al.* [YTS*11] or Denes *et al.* [DMAM19]). Thus, the input for reprojection (the previous frame) will have been partly created by reprojection as well, allowing pixels to survive as long as they are accurate enough (see Section 5.2). For pixels that have been subject to hole filling, the world position buffer is not filled and, consequently, those pixels are never warped to subsequent frames (the point sprite's vertex will be clipped). Note that, to increase performance, point sprites reprojected to the new frame's foveal region are discarded at this stage, leaving a fovea-sized hole to be redrawn later.

5. Identifying Missing Information

Reprojection does naturally not guarantee to find a one-to-one mapping for each pixel in the next frame. Consequently, there are holes with information missing afterwards. As we reproject in the periphery only, we try to fill these holes by using information present in their vicinity (see Section 5.1). We introduce a confidence function that expresses the adequacy of the reprojected and potentially hole-filled information from previous frames in Section 5.2.

5.1. Hole Filling

Using a ray tracing approach [WRK*16], hole filling is easily done by resampling the holes with adaptive ray generation. For our

rasterization-based approach, this would be equivalent to redrawing large and irregular parts of the scene, impacting performance heavily. Thus, we aim to cut down on redrawing as much as possible. For this, we use hole filling via information extrapolation. This way, especially small holes can be filled with slightly inaccurate information without being noticeable in the periphery. We use a push-pull filter [GGSC96, SGEM16] which works by creating a mip-map pyramid of existing colours (and depths with *max*) which is then used to fill holes at a mip-map level chosen relative to the hole's size. The quality of the filled hole is fed into our confidence function as well (see following section), thus hole-filled pixels are resampled after all in case their quality is not sufficient.

5.2. Confidence Function

Our confidence function is comprised of three simple terms, described in the following. Each term captures different aspects derived from the properties of the eye's receptors. The first factor, *eccentricity confidence*, is based on the falloff in the eye's visual acuity (spacial distinguishability) with larger eccentricity [Adl65], due to the lower receptor density distribution mapped to peripheral vision. This is the property that foveation methods [GFD*12, PSK*16, WRK*16, MDZV18] generally exploit, having especially the cones in mind [PSK*16]. We use

$$c_e(\phi) = S_1\left(\frac{\phi - \phi_F}{\phi_0 - \phi_F}\right),$$

with S_1 the smooth-step function, ϕ the eccentricity of the current sample, ϕ_F the angle covered by the fovea and ϕ_0 the angle we expect to be fully into the periphery. Therefore, our confidence function, by construction, rapidly converges to zero when approaching the foveal region. As shown in Figure 4 (red graph in relation to the cone distribution in blue) our parameter setting is $\phi_F = 10^\circ$ and $\phi_0 = 30^\circ$. Note that even though the eye has finer granularity in cone receptors densities than our graph suggests, as well as a blind spot in the visual field, the precision of our eye-tracking hardware did not suffice to exploit these features. With more reliable gaze information, we are convinced this can be included in the eccentricity falloff function to further decrease ambiguous areas.

The second factor, *contrast confidence*, is relevant for pixels that were subject to hole filling. Since the periphery, where eccentricity confidence is high, is mostly composed of rods which are highly sensitive to brightness, contrast shapes are easily perceived in the periphery and need to be similar to their actual appearance in our method. Reprojection itself does not introduce blur, but the push-pull filter produces slight inaccuracies. Therefore we rate the confidence for hole-filled high contrast areas (high luminance difference) lower, as seen in Figure 4 (the green graph in relation to the black rods distribution). This factor is computed along with the mip-map for the push-pull filter, where the alpha-channel holds the confidence. For each interpolation step, we set the confidence of the mip-mapped pixel to

$$c_c(p_0, \ldots, p_3) = 1 - 0.5(\max(L_0, \ldots, L_3) - \min(L_0, \ldots, L_3)),$$

where p_i are the the four contributing input pixels and L_i is the pixel's luminance.

The third factor, *hole-size confidence*, also pertains to pixels resulting from hole filling. It is a factor that lowers the confidence with respect to the size of the hole that was filled to obtain a value for the pixel in question, i.e. $c_s(m) = A^m$, where m is the mip-map level used to fill a hole and we empirically found $A = 0.6$ to be a good fit. This confidence attenuation catches two classes of artefacts that can arise due to hole filling: Firstly, holes are filled with an extrapolated colour which can change if the hole size changes in subsequent frames. This can result in high frequency colour changes in the reprojection (which is easily perceptible by rods [Adl65]). Secondly, larger holes are more likely caused by disocclusions, which we need to redraw to avoid objects popping into view once the gaze direction changes.

We tested incorporating additional physical factors into the confidence function, such as the wider distribution of receptors sensitive to blue light or decreased differentiation of red-green colours in the outer regions, but this showed to be a costly option in terms of performance, with little perceptual benefit. This is likely the case because colour- and receptor-based differentiation is implicitly factored in via the usage of luminance for our contrast confidence.

Our computed confidence value, $c = c_e(\phi)c_c(p_0, \ldots, p_3)c_s(m)$, allows us to catch several problematic configurations that can arise from reprojection and hole filling. Unless otherwise stated, we use a threshold $\varepsilon = 0.2$ (see evaluation in Section 8 for details on the confidence threshold) as the cutoff between keeping the reprojected/synthesized pixel and redrawing it. The result of this decision (the value 1 for 'needs to be redrawn' and 0 'keep as is') is stored, per pixel, in the redraw map.

6. Image Generation

Based on the redraw map, it is clear which pixels have to be recomputed. In the following, we describe how this is accomplished in an efficient manner (Section 6.1) and how the thusly recomputed data is composed with the reprojected data to provide a stable input for the HMD (Section 6.2).

6.1. Culling and Redrawing

Our *foveation culling* scheme based on the redraw map is two-fold: It works by discarding geometry in a prepass before the vertex stage (similar to z-culling) where bounding boxes are completely covered by sufficient information as well as by discarding pixels after rasterization (similar to early z-testing) to further thin out the data.

To cull geometry, inspired by hierarchical z-culling [GKM93, ZMHHI97], we compute a mip-map of the redraw map (using the *or* operator). This allows us to efficiently query larger screen-space areas for whether there is any pixel that needs redrawing. Using this, we can project the axis-aligned bounding boxes of our scene's objects to screen space and query the redraw map at the level corresponding to the object's screen-space footprint. This query at the four bounding box corners returns if any pixel of the object needs to be redrawn or if the whole object can be culled.

The models in our scenes are subdivided into objects (submodels); for us at most 500 faces per object provided the best results.

Additionally, all objects are set up for indirect drawing [EM14]. Using this with command buffers [Wor16] allows cheap and efficient culling of objects, as render buffers can be manipulated directly on the GPU. Thus, each compute shader invocation tests whether a submodel needs to be redrawn and flags or unflags the submodel for rendering.

After that, rendering proceeds with pixels not marked for redrawing being discarded both while creating the G-buffer and in the shading step. The first part causes the G-buffer to only be filled where output pixels are necessary, the second one ensures that no unnecessary shading computations are done.

6.2. Final Composition

With the partially redrawn image as well as the reprojected image at hand, we can then compose the final image. When doing so, the redrawn pixels are also subjected to temporal anti-aliasing [Kar14] (redrawing is done with a jittered viewport). We also apply a motion smoothing pass for the reprojected pixels. This is necessary as small head movement or sensor noise will cause point sprites to change their pixel coordinates temporally (and not uniformly) while rasterizing the reprojection, thereby introducing swirling in the periphery. The motion smoothing exploits the previously mentioned inaccuracy of depth-based reprojection (see Figure 3), as it uses the new depth and pixel coordinate to reproject to the previous frame and looks up the pixel's history colour. Thus, only point sprites prone to flickering between pixel footprints get a different history sample, which is then used in the same way as with TAA to smooth the pixel, with colour box clipping and accumulation with $\alpha = 0.1$. The motion smoothing pass using this implicit noise generation in the shader follows the same execution path as the TAA pass, thus there is no performance impact compared to TAA of regular rendering methods. It is important to note that the input for the next frame's reprojection pass must be without TAA/motion-smoothing, as the slight blur introduced would be propagated and accumulate to noticeable artefacts after dozens of frames.

7. Dynamic Objects and Post-Processing

While the steps described above complete the basic pipeline, our design allows for additional features, namely dynamic objects and post-processing passes. These can be achieved by further extending and exploiting the confidence function.

Dynamic objects are managed by having an object-unique id (stored in the world position buffer's W-component), which is used to retrieve the change in position in this frame for the object. Then each point sprite associated with the object is moved based on this change while reprojecting to its new, corresponding position (holes left behind are handled the usual way). This point sprite based version of the dynamic object is an approximation of the actual object and especially rotations can lead to a distorted appearance over a few frames. As such, we adjust the general confidence by a fourth factor, an empirically chosen *dynamic confidence* $c_d(t) = D^t$, where t is the time in frames since the last redraw of the pixel and D is an empirically chosen factor of 0.75. Therefore, the confidence for pixels reprojected from dynamic objects

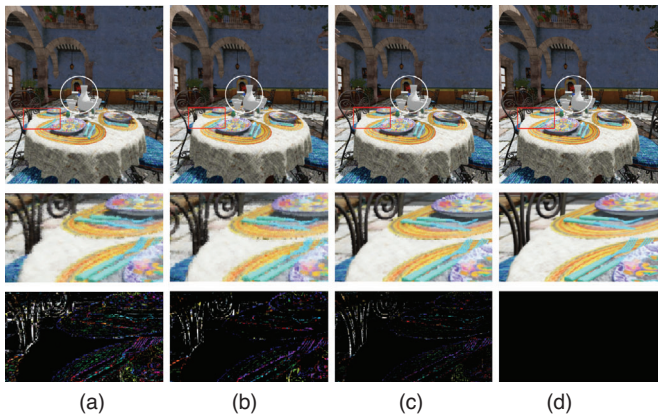

Figure 5: *Different confidence thresholds: (a) $\varepsilon = 0.05$. (b) $\varepsilon = 0.1$. (c) $\varepsilon = 0.2$. (d) Non-foveated. The white circle indicates the foveal region. Note that details (fork, knife and chair) are fuzzier with lower ε, as emphasized with the difference map between foveated and regular rendering (bottom).*

is $c = c_e(\phi)c_c(p_0, \ldots, p_3)c_s(m)c_d(t)$. The newly introduced term lowers confidence over time for pixels belonging to dynamic objects, eventually enforcing a redraw (in contrast to pixels of static objects, which will only be redrawn if the reprojection becomes inaccurate unrelated to the pixel's age).

Our method also supports acceleration for post-processing passes, which in general aim to increase visual quality after rendering and can be quite costly. Our foveated reprojection scheme re-uses final colours from previous frames, therefore there are areas in the image that do not need the recomputation of post-processing effects. In our case, we re-use tonemapping [RSSF02] and screen-space ambient occlusion [Mit07, BSD08] results implicitly by reprojecting final colours. By using the redraw map to skip computations for pixels already covered by reprojected data, we accelerate the passes considerably. Note that, usually, ambient occlusion is blurred and not computed at full resolution, thus the redraw map is sampled at a lower mip-map level to account for the larger relative screen space area of SSAO pixels.

As an example for post-processing, we chose SSAO as it provides a large visual impact and poses difficulties due to its tendency to be computed at different resolutions (which we solve by the mip-mapped redraw map). For different post-processing passes, manipulating the redraw map (or the confidence function itself) provides a clear way for inclusion into the pipeline, i.e. for lens flares to include confidence function constrains to force redrawing of bright areas even more often or for bloom to extend redraw areas by the blur radius size. While we technically accelerate TAA (as we only compute it for redrawn pixels) we included our motion smoothing pass in the same step, thus resulting in virtually no difference in overall pass performance (see Section 8.4).

8. Evaluation

In this section, we first evaluate the visually important factors of our time-warped foveated rendering algorithm and present our

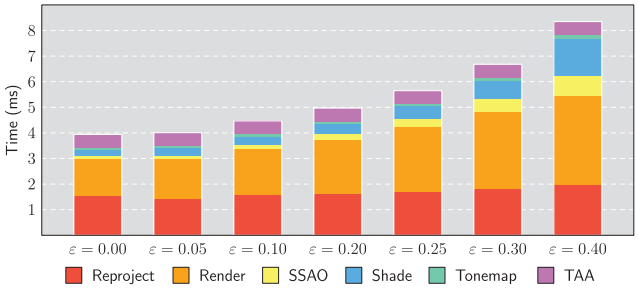

Figure 6: *Performance with different confidence thresholds: With higher thresholds, computation times (in ms) increase steeply in every category except reprojection (Times measured for Figure 1 with an Nvidia RTX 2070).*

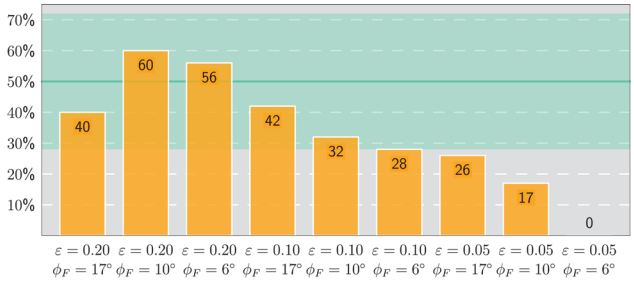

Figure 7: *Results of the calibration 2AFC user study, in percent of choices (in orange) for our foveated rendering method with different parameter sets (ε: confidence threshold, ϕ_F: fovea angle) compared to regular rendering. The green area indicates the 95% confidence interval of the binomial test.*

conducted calibration and verification user studies. Then, based on the parameter set best suited for the algorithm, we evaluate the performance against regular, non-foveated rendering, as well as foveated variable rate shading [PSK*16, Bho18]. To conclude, we present limitations posed by our algorithm as well as interesting directions for future work.

8.1. Visual Quality

Visual quality of our approach is highly dependent on the aforementioned confidence threshold ε. Confidence is computed on a scale from zero (must be redrawn) to one (good to display). As an extreme case, if ε is set to zero, no redraws would happen, thus every hole in the data is filled with extrapolated, inaccurate colours, impacting visual quality drastically. In contrast, $\varepsilon = 1$ always forces a redraw of the whole image every frame, losing any performance benefit of our foveated rendering.

For a showcase of the visual quality see Figure 5, where peripheral cutouts with different thresholds are displayed. In general, lower ε leads to more push-pull holefilling, thus fuzzier edges at objects. But while $\varepsilon = 0.05$ displays this strongly (with especially the chair loosing much of its sharp edges), with $\varepsilon >= 0.1$, the distinction is less visible.

Regarding performance, a higher confidence threshold increases computation times strongly (as seen in Figure 6), as it causes more pixel redrawing. Apart from increased shading cost, our foveation-based culling method is not able to cull as many objects, negatively impacting rendering performance. Thus, confidence thresholds higher than $\varepsilon = 0.25$ are deemed to have too high of an impact on performance for no distinct visual improvement, while $\varepsilon = \{0.05, 0.1, 0.2\}$ are good parameters to investigate perceived quality with.

8.2. Calibration User Study

To better evaluate perceived quality of our foveation we conducted two user studies. We based our study design on previous VR user studies, such as those by Patney *et al.* [PSK*16] and Fink *et al.* [FHMV*19]. With the first study, we try to calibrate our method with a suitable parameter set of confidence threshold ε and fovea angle ϕ_F.

Hardware. For the study, we used a FOVE [FOV17] HMD, a VR headset with integrated eye tracker, and ran the accompanying driver for gaze tracking calibration, followed by a manual verification of gaze tracking correctness. Rendering was done on a mobile setup with lower geometric complex scenes (low-poly *San Miguel* and a *bedroom* scene, see Figure 8(a)). During the study, frame rates were monitored and frame drops did not occur.

Study Design. Before the study, participants were only told the setup of the study (use of a VR headset, eye tracking calibrations and multiple decisions) but no specifics, and only subjects with no background in computer graphics were chosen. For the main part of the study, we used a *two-alternative forced choice (2AFC)* test method between the scene rendered regularly and in a foveated manner. Participants were asked to decide for one of the options based on visual factors of their choosing (which mode they preferred). The participants were allowed to freely switch between the two options. When switching, a gray fade was displayed for half a second. The parameters used in the setups were confidence threshold ($\varepsilon = 0.2, \varepsilon = 0.1, \varepsilon = 0.05$) and fovea angle ($\phi_F = 17^\circ$, $\phi_F = 10^\circ$, $\phi_F = 6^\circ$) in full factorial design (9 decisions per person).

Before and after the 2AFC test block, a scene was randomly shown either rendered regularly or with foveation (with parameters $\varepsilon = 0.1$ and $\phi_F = 10^\circ$, previously thought of as a good compromise), with the other setup being used after the 2AFC test. Both times, participants were encouraged to take their time and rate the scene visually on a scale from 1 to 10 based on factors of their own choosing.

Results. The study included 21 participants (12 male, nine female) between 23 and 35 years ($\mu = 27.48$, $\sigma = 2.86$), which reported a mix of previous VR experiences (seven never before, eight infrequent, six often), resulting in 20 valid rating pairs and 170 choices (one rating pair and 19 choices were discarded due to eye-tracking malfunctions or failure to look at both options).

Table 1: *Overview of the six tests conducted in the verification user study. For each test and setup (A/B), the table reports rendering modes (foveated/regular) for static and dynamic objects, scenes used, and indication of SSAO usage. Non-changing modes between setups are underlined. Participants were asked to choose between setup A and B on each scene per test, resulting in 12 decisions.*

		setup A		setup B			
		static objects	dynamic objects	static objects	dynamic objects	scenes	AO?
main tests	I - static	foveated	–	regular	–	*SunTemple1* & *Rungholt1*	×
	II - static & dynamic	foveated	foveated	regular	regular	*BistroInterior* & *LostEmpire*	×
	III - static, with AO	foveated	–	regular	–	*SunTemple1* & *Rungholt1*	✓
	IV - static & dynamic, with AO	foveated	foveated	regular	regular	*SunTemple2* & *Rungholt2*	✓
support tests	V - foveated vs. mixed	foveated	foveated	foveated	regular	*BistroInterior* & *LostEmpire*	×
	VI - mixed vs. regular	foveated	regular	regular	regular	*BistroInterior* & *LostEmpire*	×

The results of the 2AFC test are shown in Figure 7. The confidence threshold showed to have a bigger impact on preference than the fovea angle. Tested for significance (binomial test with 95% confidence), only in setups with parameters set to $\varepsilon = 0.05$, a clear preference for regular rendering can be seen (percent of choices fall outside the confidence interval). For the other options, no clear preference is indicated by the data (all lie inside the confidence interval), which is consistent with comments made by participants after the study, as many reported only being able to find differences in about half the options. On average more than five switches between options were made before a decision was reached, indicating an indecisiveness in the choices.

The second test (rating the scenes) was designed to test if artefact detection could be learned through our 2AFC test. Of our participants, 11 were randomly shown the foveated scene first, the other nine the regular. Both groups rated the visual quality of the first shown scene similar: Foveated-first with $\mu = 6.36$ ($\sigma = 1.5$) and regular-first with $\mu = 6.33$ ($\sigma = 2.0$). After the 2AFC test, the foveated-first group rated the regular rendering as the second scene on average higher by 0.45, while the regular-first group chose a rating 0.89 points lower for the foveated one. While the first is not statistically significant (p-value $= 0.096$), the second one is (p-value $= 0.009$) when tested using a t-test with 95% confidence. This shows, that even if visual quality of our foveated method is perceived similar in isolation, the differences found can impact perceived quality negatively.

We conclude from our calibration study that confidence threshold set to the lower limits seems to have impact on preference. From the remaining parameter sets, we chose $\varepsilon = 0.2$ and $\phi_F = 10^\circ$ as a conservative parameter set. Furthermore, we suggest using such a parameter set (even though the impact on performance), as we saw tendencies that users would negatively rate foveated renderings once differences through foveation were noticed.

8.3. Verification User Study

To verify the parameter set found in the first study, we conducted a second user study. It is designed to provide proof for the following three claims we presented in this paper:

1. Our parameter set *generalizes* on different kinds of static scenes and VR hardware.
2. Our handling of *dynamic* objects with dynamic falloff is sufficient.
3. *Ambient occlusion* is reprojectable in our method, even with dynamic objects.

Hardware. We chose to use the HTC Vive Pro Eye [HTC19], another VR headset with eye tracking hardware. Rendering was done on an NVIDIA RTX2070 and the frame rate limit of 90Hz was hit throughout the study in all setups.

Study Design. We recruited a new set of participants for the second study, and they were again only told its outlines. The verification user study followed the same paradigm as the main part of the calibration user study (2AFC, decision for which mode participants preferred) and was composed of six tests, four main tests and two supporting tests (see Table 1 for reference). Each of the tests was conducted on two different scenes, respectively, giving a total number of twelve 2AFC decisions for each study participant. The pairs of scenes used for each test consisted of one more realistic and one Minecraft-style scene, as the latter should provide a stronger challenge for our method due to the prevalence of sharp edges. Parameters for our method were set up as discussed before ($\varepsilon = 0.2$ and $\phi_F = 10^\circ$). For all tests, participants were asked to switch back and forth between two setups (see columns "setup A" and "setup B" in Table 1). Participants were encouraged to take their time and to choose the visually more pleasing mode. After each decision, participants were also asked to rate both setups on a scale from 1 to 10.

The first test (I) comprised *only static* scenes (*SunTemple1* and *Rungholt1*, see Figure 8(b)), with participants switching between our method and regular rendering. This enabled direct testing of the parameters found in the calibration user study on new scenes.

To test our method on *dynamic content* (test II) and thus support the second claim, dynamic objects of varying speeds and distances to the user were added to two other scenes (paper planes on *BistroInterior* and pigs on *LostEmpire*, see Figure 8(c)). In this test, participants were again asked to compare our method extended with dynamic objects (as described in Section 7) against regular rendering.

To better understand the effect of dynamic objects on our method, we also did two *supporting tests* with the same scenes. In them we fused foveated and regular rendering in one setup, rendering static

Figure 8: *Scenes used in the user studies: (a) San Miguel (top, low poly) and Bedroom (bottom), used for the calibration study. (b) SunTemple1 (top) and Rungholt1 (bottom) used for parameter verification and SSAO. (c) BistroInterior (top) and LostEmpire (bottom) used for assessing dynamic objects. (d) SunTemple2 (top) and Rungholt2 (bottom) used for our full feature set. Models downloaded from Morgan McGuire's Computer Graphics Archive [McG17] and the Nvidia ORCA library [Epi17, Lum17].*

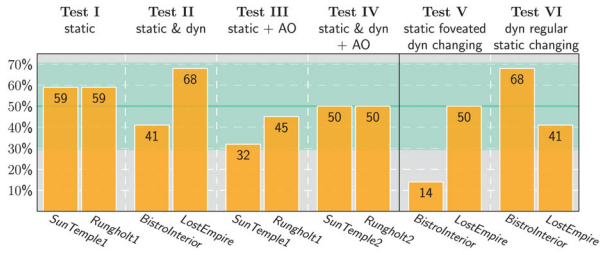

Figure 9: *Results of the verification user study, in percent of choices for setup A (foveated) as seen in Table 1. The green area indicates the 95% confidence interval of the binomial test.*

objects in a foveated fashion and dynamic objects regularly (called the *mixed mode* in the rest of the paper). One supporting test (V) was set up comparing all-foveated rendering with this mixed mode (thus only changing the rendering mode of dynamic objects), the other test (VI) examined the mixed mode set up against all-regular rendering (only switching modes for static objects). Both supporting tests should provide insights if perception of foveation changes with dynamism in scenes and if coherency in the rendering modes is preferred.

The last two main tests included SSAO (as described in Section 7), once without dynamic objects (test III, on *SunTemple1* and *Rungholt1*, see Figure 8(b)) and once with dynamic objects (test IV, on *SunTemple2* and *Rungholt2*, see Figure 8(d)). The task was to compare fully foveated and fully regular renderings in these tests.

Results. The study included 22 participants (17 male, five female) between 20 and 27 years ($\mu = 23.73$, $\sigma = 2.97$), with the majority having no prior VR experience (12 never before, seven infrequent, three often).

The results of the 2AFC tests are shown in Figure 9, showcasing no significant difference in choices for the four main tests. Between 32% and 68% chose our method (with 50% being the expected random guess value), which lies in the confidence interval for our N. In setup rating no significant difference is present. Also no significant impact of scene type (realistic or Minecraft) on decisions could be identified (p-value $= 0.091$)

The oddity in the data is the test on the *BistroInterior* scene in test V (the first support test; column 9 in Figure 9): On this particular combination of test and scene (the *LostEmpire* scene showed no significant difference, see column 10) users preferred the more detailed regularly rendered dynamic objects to our foveated version. Setup ratings also showed significant (p-value $= 0.012$) preferences in favor of the mixed mode (rendering static objects foveatedly and dynamic objects regularly) compared to rendering all foveated. The logical assumption would be that users also prefer all regular rendering to all foveated rendering on the same scene, but this claim is not supported by the data (test II, column 3) as there is no significant difference. Our interpretation of this test is that the mixed mode guides the user to notice accurately rendered dynamic objects in the periphery (as we render them regularly on-top of foveated static objects), leading to a clearer decision when comparing to all foveatedly rendered, where all of the periphery is exhibiting more fuzziness. Thus the lack of coherent render modes in the mixed mode seems to be the main reason for this strong discrepancy in choices, which clearly favor the mixed mode's more accurate dynamic objects. Our summary from this test is that once participants noticed inaccuracies in moving objects, they preferred more accuracy and thus, in some scenarios tightening dynamic confidence or using a mixed mode could provide better perceptual results. But in our suggested setups (i.e. coherent render modes), even if users might choose different constrains on dynamic objects, no significant preference for regular rendering when compared to foveated rendering is present (as seen in tests II and IV).

Takeaways. We conclude from our verification study, that for a conservatively chosen parameter set ($\varepsilon = 0.2$ and $\phi_F = 10^\circ$), generally no preference towards one of the options in our suggested feature sets could be identified. For the rest of this paper, this parameter set is used for evaluation.

8.4. Performance

In this section, we evaluate the performance of our algorithm in contrast to regular, non-foveated rendering as well as against a

Table 2: *Times in ms, rendered on an Nvidia RTX 2070 (1280 × 1440 per eye) and averaged over 1000 frames, for time-warped foveated rendering (Ours), foveated variable rate shading (fVRS) [PSK*16] and regular, non-foveated rendering. For our algorithm, the Reproj-column includes reprojection, hole-filling and confidence computation; Render includes G-buffer creation and foveation culling; and TAA includes composition and motion smoothing. Also note that fVRS uses a forward rendering pipeline, thus rendering and shading are combined in the Render-column.*

Scene	Method	Reproj	Render	SSAO	Shade	Tonemap	TAA	Overall
Bistro,	Ours	1.62	2.11	0.23	0.40	0.10	0.51	**4.97**
(Figure 1)	Regular		4.12	1.03	2.05	0.20	0.51	7.91
	fVRS		6.30			0.11	0.77	7.18
San Miguel,	Ours	1.77	4.10	0.74	0.63	0.11	0.51	**7.86**
(Figure 11(a))	Regular		7.11	1.52	2.02	0.17	0.53	11.35
	fVRS		8.72			0.13	0.72	9.57
Sponza,	Ours	1.61	0.40	0.21	0.33	0.15	0.51	3.21
(Figure 11(b))	Regular		0.95	0.83	0.67	0.18	0.51	3.14
	fVRS		0.90			0.13	0.78	**1.81**

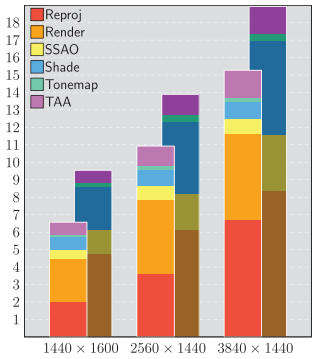

Figure 10: *Times in ms for other common HMD per-eye resolutions: Our method (front) has diminishing speedups compared to regular rendering (back, grey tone), down to about 1.25× with 8K devices.*

Figure 11: *Scenes used in the evaluation: (a) San Miguel (Designer: Guillermo M. Leal Llaguno). (b) Sponza (Designers: Marko Dabrovic, Frank Meinl). Also used: Bistro [Lum17] (Figure 1). Models downloaded from Morgan McGuire's Computer Graphics Archive [McG17].*

state-of-the-art technique, foveated variable rate shading (fVRS) [PSK*16], which accelerates rendering by reducing the shading rate in the periphery (to 2 × 2 with eccentricities between 10 and 20 degrees and 4 × 4 beyond), which is especially efficient due to hardware support on recent GPUs [Bho18].

Setup. We evaluate the methods with a FOVE HMD (also used in the calibration user study), with a resolution of 1280 × 1440 pixels per eye, on an Nvidia RTX 2070. We used three scenes for evaluation: Crytek's *Sponza* scene (Figure 11(b)), an atrium with a low number of triangles (262K) and small texture memory footprint, *San Miguel* (Figure 11(a)), a courtyard with a high number of triangles (8.8M) including small, expensive to render leaves with alpha testing, and Amazon's Lumberyard *Bistro* scene (Figure 1), a well-balanced scene with moderately high triangle count (3.9M), similar to modern video game scenes [LA19]. For *Bistro* and *San Miguel*, one directional light and 20 point lights illuminate the scene, while for *Sponza* (as an overall less complex example) only one directional light is used.

Unless otherwise noted, our algorithm as well as the regular rendering use a deferred renderer with a 144-bit G-buffer, frustum culling, normal mapping, alpha testing and 16-bit HDR GGX shading [Kar13]. Time-warped foveated rendering additionally uses a 128-bit world-position buffer as well as the previously mentioned foveation culling. As additional post-processing, we use screen space ambient occlusion (SSAO) at half resolution with 32 samples per pixel. Foveated VRS uses a forward renderer with the same attributes as regular rendering, but with texture-baked ambient occlusion and without alpha testing, as it is not supported [PSK*16].

Results. The times for the scenes can be seen in Table 2, averaged over 1000 frames. In general, our method compares favourably on both *Bistro* and *San Miguel* as we achieve speedups of 1.6× and 1.4× compared to regular rendering. Compared with fVRS, performance gains are smaller, but still speedups of 1.4× and 1.2× are measured. On *Sponza*, our method performs considerably slower than both regular and fVRS due to scene layout: Our method has a constant overhead of around 1.7 ms for reprojecting, thus on low-cost scenes such as *Sponza*, our increased performance, which comes from geometry processing, shading and post-processing, is not as relevant. Following is a detailed evaluation of these three parts

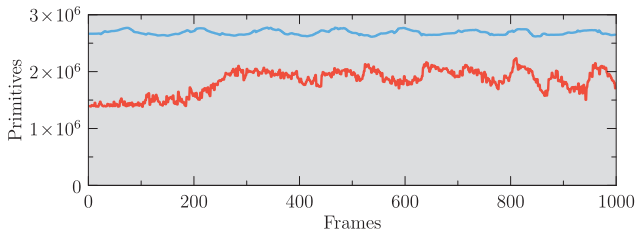

Figure 12: *Primitives over 1000 frames in the* Bistro *scene (Figure 1). Ours (red) only processes 1.8 million primitives on average, in contrast to regular and fVRS (blue) with 2.7 million.*

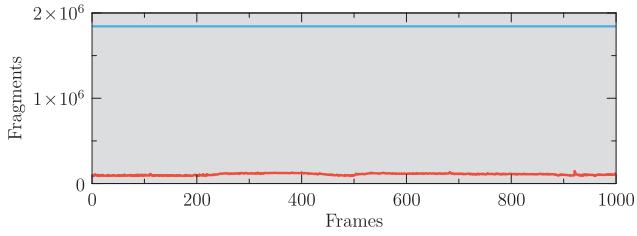

Figure 13: *Amount of fragments shaded per eye over 1000 frames: With regular rendering, all 1.84M fragments per eye (resolution of* 1280 × 1440*) need to be shaded (blue), for our approach on average 108K fragments (6%) are shaded (red).*

as well as the impact of dynamic objects and resolution on performance.

Geometry Processing. Our algorithm reduces vast amounts of geometry transformation with our foveation culling scheme. A graph of this can be seen in Figure 12, which shows that our method cuts down primitives by roughly one third on the *Bistro* scene (similar as with the other scenes, with reductions of 27% on *San Miguel* and 35% on *Sponza*). This is vital for our method's performance, as without it, creating the G-Buffer (see *Render* column in Table 2) would only be slightly faster than with regular rendering through the ability to discard pixels in confident areas. It is important to note that the cost of performing the culling itself (as described in Section 6.1) is consistently very low at about 0.02 ms in our scenes (included in the *Render* column in Table 2).

Shading. We use the redraw map to discard pixels already reprojected and exhibiting a confidence value above the threshold, thus reducing the amount of shaded pixels drastically. An overview of this can be seen in Figure 13. On average (computed over 1000 frames), we shade only 108K fragments ($\sigma = 10.6$K) per frame, a reduction of 94% in contrast to regular, full resolution rendering and exceeding fVRS' reduction of 75%. Thus our method achieves great shading performance in complex shading scenarios (speedups of 5.1× (*Bistro*) and 3.2× (*San Miguel*), see column *Shade* in Table 2) while also reducing computation times in simple shading setups by about half (*Sponza*).

Post Processing. We can avoid recomputing post-processing passes (such as SSAO) for reprojected pixels, resulting in good

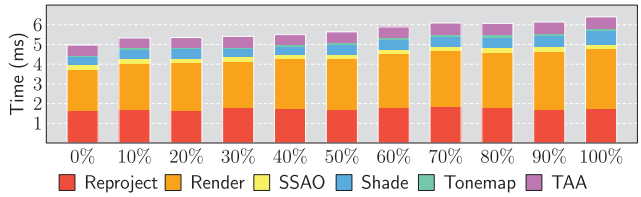

Figure 14: *Computation time in relation to total footprint of dynamic objects on screen: With a larger dynamic object footprint, computation times increase noticeably. In the worst case (100% dynamic object screen footprint) our method slows down by about 25%.*

speedups of 2.1× to 4.5× based on scene complexity. We chose to not compare a fVRS-adjusted SSAO implementation with our algorithm, as it would require vast changes to a lower-resolution SSAO pass to fit with different shading rates efficiently and would diverge far from the original publication. For colour adjustment passes (e.g. used here: tone mapping with fixed luminance), recomputations can also be skipped, with possible speedups of around 1.2× to 2.0×. However these reductions are less significant due to the low overall cost of these passes.

Dynamic Objects. Our method handles inaccuracies stemming from reprojecting dynamic objects by lowering confidence with the dynamic confidence falloff. This forces dynamic objects to be redrawn more often on average. The confidence computation is done on pixels, thus the screen-space size of dynamic objects is of relevance here. The performance impact can be seen in Figure 14, with times for different amounts of dynamic falloff present in the scene. Generally, the more pixels are covered by dynamic objects, the more pixels are necessary to be redrawn, thus all parts of our algorithm except reprojection take longer to compute, with our method's computation time increasing by about 25% in the worst case (100% of pixels subject to dynamic confidence falloff).

Render Resolution. So far, our method's performance was evaluated using the FOVE's resolution. For times comparing different resolutions of our method and regular rendering, refer to Figure 10. With HTC Vive Pro Eye's resolution of 1440 × 1600 pixels per eye, our overall speedup is in a similar range with 1.45×, while for higher targeted resolutions the overall speedup decreases to only about 1.25× with 8K devices (3840 × 1440 pixels per eye). This is due to our reprojection computation cost scaling almost linearly with resolution, taking an overall larger percentage of the render times with higher resolution settings. It it important to note however that higher-resolution HMDs usually target a larger field of view than the FOVE or HTC Vive Pro Eye, thus we are optimistic that our method (and especially the confidence function) can be tuned to reduce rendering cost in the far peripheral regions further (due to even lower receptor densities), once high-resolution, high field-of-view HMDs with eye trackers are available.

Performance Conclusion. Our method provides a solid performance increase with complex scenes, as shown with the *San Miguel* and *Bistro* scenes. Foveation culling allows our redrawing pass to

ignore large amounts of geometry and the shading rate is reduced to below 10% through redraw-map-based pixel discards, especially promising due to the overall high shading cost of modern video games [SG16, LA19]. Furthermore, post-processing passes such as SSAO also profit from the decrease in pixels to be processed. This results in speedups for the pipeline of up to 1.6×, which allows the additional time to be put towards increasing visual quality or using more complex setups in VR (such as *San Miguel*, which is regularly rendered over the 90 Hz/11.1 ms VR threshold).

8.5. Limitations and Future Work

Our algorithm is subject to a small but significant set of limitations: Firstly, transparency presents a problem not solvable by our approach, it is only possible to add a transparent pass after our pipeline finishes (resulting in no speedup for transparent objects compared to regular rendering). Transparent fragments lack a distinct position after rendering (as they are commonly blended), thus reprojecting will yield wrong results under movement. We recommend following Schollmeyer *et al.*'s [SSB*17] A-buffer method if translucent materials are to be reused.

Secondly, moving lights are not supported in our method, as they drastically change shading of fragment between frames. Though it should be possible to heuristically evaluate noticeability of colour changes in fragments and thus lower confidence in them, this will probably have a considerable impact on performance due to the possibly high numbers of pixels to redraw.

Thirdly, view-dependent post-processing effects (e.g. reflections) may present problems. Even though our scenes heavily use specular highlights, our consideration of brightness in the confidence computation eliminates strong discrepancies in highlights through head or gaze movement. However, we cannot rule out that reprojected reflections can cause distracting artefacts, thus, in its current form, for our time-warped foveated rendering method, reflections must be seen as a limitation and an interesting opportunity for future work.

As for additional future work, our verification user study (as seen in Section 8.3) provides hints that perception of dynamic objects in the periphery can still be improved. For that, extending the confidence function with more dynamic-focused perceptual terms would be an interesting avenue for future work, both algorithmically and from a perceptual modelling standpoint, as our method's performance is impacted noticeably by scene dynamism.

9. Conclusion

With this paper we have presented a novel time-warped foveated rendering technique well suited for current HMD hardware with integrated eye tracking. While it lacks the simplicity in implementation compared to established work [PSK*16, Bho18], our algorithm outperforms them on scenes similar to real-world use cases and introduces an effective perceptual formalization of reprojection quality allowing a larger feature set. Though our method fails in some common scenarios such as transparency, moving lights and reflections, we believe that our approach strikes a promising compromise between performance and features, well suited for modern real-time VR rendering.

Acknowledgements

Open access funding enabled and organized by Projekt DEAL.

References

[Adl65] Adler F. H.: Physiology of the eye. *Academic Medicine 40*, *7* (1965), 720.

[APLK17] Albert R., Patney A., Luebke D., Kim J.: Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP) 14*, *4* (2017), 25.

[Bho18] Bhonde S.: NVIDIA Developer Blog: Turing Variable Rate Shading in VRWorks, 2018. https://devblogs.nvidia.com/turing-variable-rate-shading-vrworks/.

[BMS*12] Bowles H., Mitchell K., Sumner R. W., Moore J., Gross M.: Iterative image warping. *Computer Graphics Forum* , *31*, (2012), 237–246.

[BSD08] Bavoil L., Sainz M., Dimitrov R.: Image-Space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks* (2008), ACM, p. 22.

[CSKH90] Curcio C. A., Sloan K. R., Kalina R. E., Hendrickson A. E.: Human photoreceptor topography. *Journal of Comparative Neurology 292*, *4* (1990), 497–523.

[dCI17] de Carpentier G., Ishiyama K.: Decima engine: Advances in lighting and aa. *SIGGRAPH, Advances on Real-time Rendering Course* (2017).

[DMAM19] Denes G., Maruszczyk K., Ash G., Mantiuk R. K.: Temporal resolution multiplexing: Exploiting the limitations of spatio-temporal vision for more efficient VR rendering. *IEEE Transactions on Visualization and Computer Graphics 25*, *5* (2019), 2072–2082.

[DRE*10] Didyk P., Ritschel T., Eisemann E., Myszkowski K., Seidel H.-P.: Adaptive image-space stereo view synthesis. In *15th International Workshop on Vision, Modeling, and Visualization* (2010), Eurographics Association, Darmstadt, pp. 299–306.

[DWWL05] Dayal A., Woolley C., Watson B., Luebke D.: Adaptive frameless rendering. In *ACM SIGGRAPH 2005 Courses*. 2005, pp. 24–es.

[EM14] Everitt C., McDonald J.: Beyond Porting— How Modern OpenGL Can Radically Reduce Driver Overhead, 2014. https://developer.nvidia.com/content/how-modern-opengl-can-radically-reduce-driver-overhead-0.

[Eng14] Engel W.: Per-pixel lists for single pass A-buffer. In *GPU Pro 5*. AK Peters/CRC Press, Natick, MA, 2014, pp. 18–25.

[Epi17] Epic Games: Unreal engine sun temple, open research content archive (orca), October 2017. http://developer.nvidia.com/orca/epic-games-sun-temple.

[FHMV*19] Fink L., Hensel N., Markov-Vetter D., Weber C., Staadt O., Stamminqer M.: Hybrid mono-stereo rendering in

virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)* (2019), IEEE, Piscataway, NJ, pp. 88–96.

[FOV17] FOVE: FOVE Specifications, 2017. https://www.getfove.com/.

[FRS19] Friston S., Ritschel T., Steed A.: Perceptual rasterization for head-mounted display image synthesis. *ACM Transactions on Graphics (TOG) 38*, *4* (2019), 1–14.

[GB16] Goldstein E. B., Brockmole J.: *Sensation and Perception*. Cengage Learning, Boston, MA, 2016.

[GFD*12] Guenter B., Finch M., Drucker S., Tan D., Snyder J.: Foveated 3D graphics. *ACM Transactions on Graphics (TOG) 31*, *6* (2012), 164.

[GGSC96] Gortler S. J., Grzeszczuk R., Szeliski R., Cohen M. F.: The lumigraph. *SIGGRAPH*, *96*, (1996), 43–54.

[GKM93] Greene N., Kass M., Miller G.: Hierarchical Z-buffer visibility. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), ACM, pp. 231–238.

[HMT18] Hoffman D., Meraz Z., Turner E.: Limits of peripheral acuity and implications for VR system design. *Journal of the Society for Information Display 26*, *8* (2018), 483–495.

[HTC19] HTC: HTC Vive Pro Eye Specifications, 2019. https://www.vive.com/de/product/vive-pro-eye/.

[Kar13] Karis B.: Real shading in unreal engine 4. *Proceedings of Physically Based Shading Theory Practice 4* (2013).

[Kar14] Karis B.: High-quality temporal supersampling. *Advances in Real-Time Rendering in Games, SIGGRAPH Courses 1* (2014), 1–55.

[LA19] Lejdfors C., Aguaviva R.: Advanced Graphics Techniques Tutorial: Efficient Rendering in 'The Division 2', 2019. Games Developers Conference. https://schedule.gdconf.com/session/advanced-graphics-techniques-tutorial-efficient-rendering-in-the-division-2/864612.

[LKE18] Lee S., Kim Y., Eisemann E.: Iterative depth warping. *ACM Transactions on Graphics (TOG) 37*, *5* (2018), 177.

[Lum17] Lumberyard Amazon: Amazon Lumberyard Bistro, Open Research Content Archive (ORCA), 07 2017. http://developer.nvidia.com/orca/amazon-lumberyard-bistro.

[McG17] McGuire M.: Computer Graphics Archive, 07 2017. https://casual-effects.com/data.

[McM97] McMillan L.: *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, Citeseer, 1997.

[MDZV18] Meng X., Du R., Zwicker M., Varshney A.: Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, *1* (2018), 5.

[Mit07] Mittring M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 Courses* (2007), ACM, pp. 97–121.

[MKRH11] Mantiuk R., Kim K. J., Rempel A. G., Heidrich W.: Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM Transactions on Graphics (TOG) 30*, *4* (2011), 1–14.

[NSL*07] Nehab D., Sander P. V., Lawrence J., Tatarchuk N., Isidoro J. R.: Accelerating real-time shading with reverse reprojection caching. *Graphics Hardware 41*, (2007), 61–62.

[PSK*16] Patney A., Salvi M., Kim J., Kaplanyan A., Wyman C., Benty N., Luebke D., Lefohn A.: Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG) 35*, *6* (2016), 179.

[RSSF02] Reinhard E., Stark M., Shirley P., Ferwerda J.: Photographic tone reproduction for digital images. *ACM Transactions on Graphics (TOG) , 21*, (2002), pp. 267–276.

[SG16] Sousa T., Geffroy J.: The devil is in the details: idTech 666. *Advances in Real-Time Rendering in Games, SIGGRAPH Courses* (2016).

[SGEM16] Stengel M., Grogorick S., Eisemann M., Magnor M.: Adaptive image-space sampling for gaze-contingent real-time rendering. *Computer Graphics Forum*, *35*, (2016), 129–139.

[SHK*17] Sun Q., Huang F.-C., Kim J., Wei L.-Y., Luebke D., Kaufman A.: Perceptually-guided foveation for light field displays. *ACM Transactions on Graphics (TOG) 36*, *6* (2017), 192.

[SIGK*16] Swafford N. T., Iglesias-Guitian J. A., Koniaris C., Moon B., Cosker D., Mitchell K.: User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception* (2016), pp. 7–14.

[SSB*17] Schollmeyer A., Schneegans S., Beck S., Steed A., Fröhlich B.: Efficient hybrid image warping for high frame-rate stereoscopic rendering. *IEEE Transactions on Visualization and Computer Graphics 23*, *4* (2017), 1332–1341.

[SSP*18] Sitzmann V., Serrano A., Pavel A., Agrawala M., Gutierrez D., Masia B., Wetzstein G.: Saliency in VR: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics 24*, *4* (2018), 1633–1642.

[SvLBF09] Smit F., van Liere R., Beck S., Fröhlich B.: An image-warping architecture for VR: Low latency versus image quality. In *2009 IEEE Virtual Reality Conference* (2009), IEEE, Piscataway, NJ, pp. 27–34.

[TAKW*19] Tursun O. T., Arabadzhiyska-Koleva E., Wernikowski M., Mantiuk R., Seidel H.-P., Myszkowski K., Didyk P.: Luminance-contrast-aware foveated rendering. *ACM Transactions on Graphics (TOG) 38*, *4* (2019), 1–14.

[Vla16] Vlachos A.: Advanced VR rendering performance. In *Game Developers Conference* (2016), vol. *2016*.

[VST*14] Vaidyanathan K., Salvi M., Toth R., Foley T., Akenine-Möller T., Nilsson J., Munkberg J., Hasselgren J., Sugihara M., Clarberg P., et al.: Coarse pixel shading. In *Proceedings of High Performance Graphics* (2014), Eurographics Association, Darmstadt, pp. 9–18.

[VW16] Van Waveren J.: The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (2016), ACM, pp. 37–46.

[WDP99] Walter B., Drettakis G., Parker S.: Interactive rendering using the render cache. In *Rendering Techniques' 99*. Springer, New York, 1999, pp. 19–30.

[Wor16] Worcester M.: Command Buffers and Pipelines, 2016. https://www.khronos.org/assets/uploads/developers/library/2016-vulkan-devday-uk/2-Command_buffers_and_pipelines.pdf.

[WRHS18] Weier M., Roth T., Hinkenjann A., Slusallek P.: Foveated depth-of-field filtering in head-mounted displays. *ACM Transactions on Applied Perception (TAP) 15*, *4* (2018), 26.

[WRK*16] Weier M., Roth T., Kruijff E., Hinkenjann A., Pérard-Gayot A., Slusallek P., Li Y.: Foveated real-time ray tracing for head-mounted displays. *Computer Graphics Forum*, *35*, (2016), 289–298.

[WSR*17] Weier M., Stengel M., Roth T., Didyk P., Eisemann E., Eisemann M., Grogorick S., Hinkenjann A., Kruijff E., Magnor M., *et al.*: Perception-driven accelerated rendering. *Computer Graphics Forum*, *36*, (2017), 611–643.

[YTS*11] Yang L., Tse Y.-C., Sander P. V., Lawrence J., Nehab D., Hoppe H., Wilkins C. L.: Image-based didirectional scene reprojection. *ACM Transactions on Graphics (TOG)*, *30*, (2011), 150.

[ZMHHI97] Zhang H., Manocha D., Hudson T., Hoff III K. E.: Visibility culling using hierarchical occlusion maps. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), ACM Press/Addison-Wesley Publishing, New York, NY, pp. 77–88.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data S1

Data Video S2