Hybrid Mono-Stereo Rendering in Virtual Reality

Laura Fink[1], Nora Hensel[1], Daniela Markov-Vetter[2], Christoph Weber[1], Oliver Staadt[2], and Marc Stamminger[1]

[1]University of Erlangen-Nuremberg.* [2]University of Rostock.†

Figure 1: Visualization of the camera setup and the assembly of a hybrid stereoscopic image pair using only one image to serve for the far region (beyond 3 m) of both stereo views.

Abstract

Rendering for *Head Mounted Displays* (HMD) causes a doubled computational effort, since serving the human stereopsis requires the creation of one image for the left and one for the right eye. The difference in this image pair, called *binocular disparity*, is an important cue for depth perception and the spatial arrangement of surrounding objects. Findings in the context of the *human visual system* (HVS) have shown that especially in the near range of an observer, binocular disparities have a high significance. But as with rising distance the disparity converges to a simple geometric shift, also the importance as depth cue exponentially declines.

In this paper, we exploit this knowledge about the human perception by rendering objects fully stereoscopic only up to a chosen distance and monoscopic, from there on. By doing so, we obtain three distinct images which are synthesized to a new hybrid stereoscopic image pair, which reasonably approximates a conventionally rendered stereoscopic image pair. The method has the potential to reduce the amount of rendered primitives easily to nearly 50 % and thus, significantly lower frame times. Besides of a detailed analysis of the introduced formal error and how to deal with occurring artifacts, we evaluated the perceived quality of the VR experience during a comprehensive user study with nearly 50 participants. The results show that the perceived difference in quality between the shown image pairs was generally small. An in-depth analysis is given on how the participants reached their decisions and how they subjectively rated their VR experience.

Keywords: Virtual Reality, Stereoscopic Rendering, Depth Perception

Index Terms: Virtual Reality—Stereoscopic Rendering—Hybrid Rendering—Depth Perception;

*E-mail: laura.fink@fau.de, nora.hensel@fau.de, christoph.weber@fau.de, marc.stamminger@fau.de

†E-mail: daniela.markov-vetter@uni-rostock.de, oliver.staadt@uni-rostock.de

1 Introduction

Rendering for *Virtual Reality* (VR) requires the synthesis of an image pair at a high framerate. Depending on scene complexity, this can be challenging not only for low-budget gadgets, but also for modern high-end hardware. The similarity of the stereoscopic image pair has led to a number of acceleration approaches to reduce the computational costs that comes along with rendering nearly the same objects twice from only slightly different perspectives. It can easily be shown that the difference between the stereoscopic images decreases quickly with distance. Consequently, the importance of binocular disparities for depth perception drops quickly in the distance. Literature reports that human stereoscopic vision only delivers significant depth cues up to ten meters, but these numbers vary [2, 20].

A simple acceleration approach is thus to render distant objects once, and to render only close objects separately for both eyes on top. This idea can be directly applied to flight simulator scenarios [5], where the outside world is nearly identical for left and right eye and thus rendered only once, whereas the cockpit is rendered stereoscopically on top. However, in general setups it is more difficult to distinguish between close and distant objects and to achieve consistent renderings without transition artifacts.

In this paper, we examine an approach that uses a pre-defined distance m to switch between monoscopic and stereoscopic rendering. We call the plane with $z = m$ the *mono plane*, everything in front of or behind this plane is the *near region*, or *far region*, respectively. We first render the near region for left and right (up to the depth of the mono plane $z = m$) as shown in Fig. 1. Next, we render the far region using the third frustum, and use the result to fill in the remaining image parts of the left and right view. If the result is properly mapped to the left and right image, the images are continuous, and the transition is not noticeable. Note that by first rendering the near region(s), we can use the depth buffers of left and right image for occlusion culling when rendering the far regions. Depending on the choice of m, the costs for the additional rendering pass and copy operations are compensated by the reduced workload, resulting in a significant speedup. We will evaluate this effect in Sect. 5.

This general idea is not new, for instance it was implemented in the Oculus SDK and described in blog posts, e.g. [22]. However, the implementation focuses on mobile applications (GearVR) and

descriptions lack detail of the implementation, e.g. how to handle arbitrary view frusta as achieved by a precise calibration, how to avoid artifacts along the transition plane, or how to handle anti-aliasing properly. Furthermore, to the best of our knowledge, the impact on human perception has not been examined yet, and how it varies with the choice of the mono plane.

In this paper, we examine this approach in detail:

- We provide technical details of a practical implementation of this idea, including the usage of asymmetric calibrated view frusta, as well as details on culling and anti-aliasing, that go beyond available implementations.
- We present results from a user study with almost 50 participants to evaluate the visual impact of the approach, and the influence of the choice of the mono plane on perception.

2 Related Work

From a hardware point of view, *multiview* [3, 8] is one answer to compensate for the higher computational costs to render stereoscopically. Other terms which mean the same or very similar techniques to multiview are e.g. *instanced stereo* [8] or *single pass stereo* [21]. Multiview approaches mainly work by conflating multiple drawcalls which only differ in view or perspective. Unnecessary communication between GPU and CPU is avoided. Thus, performance gains are mostly expected on CPU side. In mobile context, some hardware drivers also slightly cut down computations during the geometry processing. But, as the instancing of drawcalls does not reduce the workload itself, the time spend for fragment processing and the fill rate remain nearly unchanged to conventional stereo rendering [3]. A more detailed explanation and analysis of multiview alike approaches can be seen in a post at the Unity Blog by Srinivasiah [24].

Tackling the effective workload, several perception based acceleration methods have arisen especially in VR context [28]. The general idea is to specifically make the most computational effort where it really is perceived. One of these findings is the diminishing importance of binocular disparities with distance as depth cue which was published by Cutting and Vishton [2] or McCann et al. [20]. The finding suggests the idea of only one rendering partly serving as image for both eyes as sufficient.

In 2017, an implementation of the idea to render the far region only once for both eyes, was officially released [9] for mobile applications in the Unreal Engine and is referred to as *Mono Far Field Rendering* [7]. The feature coincides with an early prototype from 2014 by Tom Heath [13] which was shipped with as code sample with an outdated version of the Oculus SDK. CThe basic concept of this implementation is described for GearVR and OculusGo in the Oculus developer documentation [22] and was besides of that presented for mobile developers at the GDC 2017 by D. Di Donato, R. Palandri and R. Vance [3]. While these implementations and talks describe the general idea that is also the basis for this paper, they do not further discuss technical details that arise in a practical implementation (e.g. how to avoid shading artifacts at the transition plane or how to integrate anti-aliasing). Furthermore, the impact of the approximation on human perception is not examined, and how scene content or the choice of the transition plane influences human perception.

Converting mono to stereo content, as done for the far region of our renderings, is a concept which is well known in research for 3DTVs and movie (post-)production [1, 10, 17, 23]. Essentially, such methods work by warping pixels according to their known or assumed depth to yield an image pair with a certain degree of disparity. Many online/realtime approaches follow a two step algorithm: at first, pixels from the input image are warped according to a given mapping function and secondly, the occurring artifacts are handled [23]. The arising artifacts are mainly holes which are caused by disocclusions and corrected by thought-out filling algorithms [1, 10]. With HMDs, the comparably larger size of the perceived footprint demands consistently filled pixels and temporal stability of the hole filling results and are crucial for the perceived image quality [27]. Estimating content from the adjacencies is not trivial and results in a trade-off between performance gain and an investment of computation time for more reasonable hole filling. As performance gain is our main goal, we simply map pixels according to a constant depth throughout the mono plane. Treating the depth at the plane as constant, prevents holes from the outset but serves of course only as crude approximation for the mapping. Non-linear warping functions can circumvent the need of hole filling, too, but the selection of an appropriate warping function and the higher computational costs are rather applicable to an offline post-production use case than to realtime applications [17].

User studies emerged as commonly accepted tool to evaulate the impact of such approximations, as done e.g. by Schollmeyer et al. [23] or Lang et al. [17] in recent publications. Since we classify the alterations of the final stereoscopic image pair as similar to the ones mentioned, we followed their methods to assess the performance of hybrid mono-stereo rendering in terms of image quality and their considerations to take account for the HVS.

3 Hybrid Mono-Stereo Rendering

In the following, we assume a typical HMD stereo-setup with two parallel image planes, and an offset *IPD* (interpupillary distance) of about 6.4cm. For a hybrid renderer, we have to define a third frustum for the scene beyond the mono plane, and how to merge the images so that there is no visible transition. Because of findings about the *cyclopean eye* serving as the perceived origin of a projection line from the observer towards an object [6, 19] (instead of a dominant eye), we treat both eyes as equally important, so any introduced error should distribute symmetrically. The transition from stereo- to monoscopic image parts should be continuous and not cause a notable edge. As far as it is possible no objects should get lost or added to the visual field. At last, we make no simplification of any hardware-specified parameters, that is we want to be able to use the arbitrary, device-specific calibrated frusta, as they are used in high-end HMD devices.

In previous work [3, 7, 22], the setup shown in Fig. 2 (left) has been used. Frusta are assumed to be completely symmetric and not skewed, so for each camera the same projection matrix is utilized [22]. In our setup, we decided to position the third camera C as shown in Fig. 2 (center left) to better fulfill the above described properties. We choose the union volume of frustum L and R behind the mono plane as starting point to construct frustum C. It provides the inclusion of the visible content of both eyes. At the same time, it shares at least one aperture angle with each stereo frustum, hence a certain similarity of perspective is implied. Ideally, the placement of camera C would be the intersection point of all expanded planes of the union volume, but due to asymmetries of hardware parameters there is no intersection point guaranteed. Hence, we approximate the camera position by the rear intersection of the left and right frustum plane with the z-axis. Simultaneously, we demand that the third frustum contains all intersections of the stereo frusta with the mono plane to provide the prerequisites of a continuous transition.

Based on these triple frusta, the image formation is simple: The far region is rendered using view frustum C, with the mono plane $z = m$ as image plane. The resulting monoscopic image is then used as background for the renderings of the left and right eye (frusta L and R), which is in the simplest case (without any frustum asymmetries) a shifted copy operation of the monoscopic image. Note that a real implementation requires further considerations, e.g. to avoid artifacts at the transition due to sub-pixel translations, or to obtain proper depth values for post-processing. We will elaborate on these implementation issues in Sect. 4.

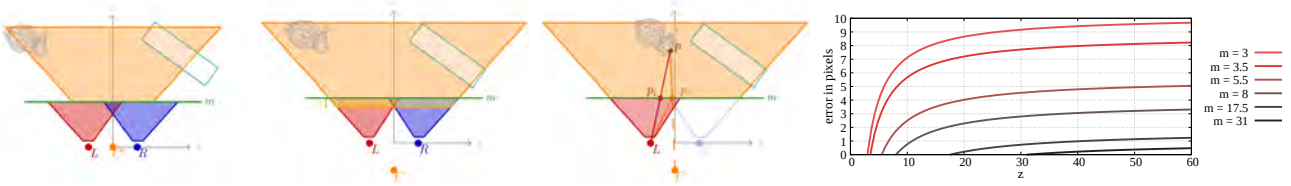


Figure 2: Left: Three camera setup for hybrid rendering of *Mono Far Field Rendering* [3, 7, 22] with mono plane m (darkgreen). Center left: Our setup for hybrid rendering with mono plane m (darkgreen) and an overlap region (ε). Note that our setup correctly contains the entire bunny. Center right: Projection error for our setup. A point p is first projected towards the camera C onto the mono plane m (p_C), then towards the left (or right) eye. Note that the interpupillary distance is vastly exaggerated to see the effect. Right: Maximal pixel offset over depth for different choices of the mono plane (HTC Vive, $w = 1344$ px (super sampled), $IPD = 0.064\,m$).

Error Analysis The plot in Fig. 2 (center right) shows how our triple setup alters the projection of a point for objects beyond the mono-plane. In our setup, projected points generally move inwards. Due to construction, the error at the mono plane is zero and grows with growing distance, which fits well with the fact that stereoscopic depth perception quickly decreases with distance. The closer the mono plane is chosen, the larger the error becomes. Fig. 2 (right) shows the maximum projection error of our setup for different choices of m. Note that even for very close m the error is in the range of a few pixels. For instance, in a practical setup with the mono plane at 8 meters the maximum error is 4 pixels.

The transition of one projection matrix to another along the mono plane leads to a continuous rendering. However, the transition plane can result in subtle kinks of lines passing this plane. The effect becomes visible for lines close to the viewer and parallel to the view direction. Note that this effect is unavoidable and also appears in previous work [3, 7, 22]. While the effect is hardly noticeable in practice, it can be seen by experienced users. We will discuss this feature in more detail in the context of our user study in Sect. 6.

4 Implementation Details

In this section, we describe technical details of our implementation, including the usage of calibrated, asymmetric view frusta, culling optimizations, the avoidance of transition artifacts and the inclusion of anti-aliasing techniques. Fig. 3 shows our pipeline, in the following we describe the single stages.

As a result, our renderer has a significant advantage in performance, but also achieves rendering quality that is on par with traditional stereoscopic rendering (except the inevitable kink which is only visible during closer examination) and includes proper anti-aliasing.

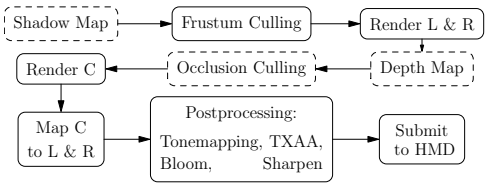


Figure 3: Pipeline of our implementation

4.1 View and Projection Matrices

Our camera setup as shown in Fig. 2 consists of the three cameras *L*eft, *R*ight and *C*enter. Each camera has an individual view matrix $V_{L,R,C}$ and projection matrix $P_{L,R,C}$. The projection values *l*eft, *r*ight, *b*ottom and *t*op, as well as viewport dimensions $w_{L,R}$ and $h_{L,R}$ are hardware-given and retrieved from the SDK [18, 25] (see Sect. 4.2 and 5 for further details). The origin of the coordinate system in Fig. 2 corresponds to the head position of the user and the z-axis to its view direction, which implies that the common view matrix V is already applied.

$V_{L,R,C}$ differ from V by $\Delta V_{L,R,C}$, which are the matrix representations of the translations t:

$$t_L = -\left(\tfrac{1}{2}d,0,0\right)^T \quad t_R = \left(\tfrac{1}{2}d,0,0\right)^T \quad t_C = (0,0,z_C)^T, \\ \text{with } z_C = \min\left(\frac{d}{2l_L}, \frac{-d}{2r_R}\right). \tag{1}$$

where d represents the interpupillary distance. So, t_C is the rear intersection of the left and right frustum plane with the z-axis. We demand that the third frustum contains all intersections of the stereo frusta with the mono plane to provide the prerequisites of a continuous transition. This is enforced by using the slope of the lines which connect the camera position and the outer intersection points. Thus, we calculate

$$\begin{aligned} l_C &= \max\left(g(m,r_L,t_{LX}, g(m,r_R,t_{RX})\right), \\ r_C &= \min\left(g(m,l_L,t_{LX}), g(m,l_R,t_{RX})\right), \\ b_C &= \min\left(g(m,t_L,0), g(m,t_R,0)\right) \quad \text{and} \\ t_C &= \max\left(g(m,b_L,0), g(m,b_R,0)\right) \end{aligned} \tag{2}$$

with $g(m,a,b) = m - {z_C}^{-1}(am + z_C) + b$.

Further, we refer to m as the distance of the mono plane from the mentioned origin. Additionally we introduce an overlap ε, its purpose is described in Sect. 4.3, with

$$\varepsilon(m) = k|s_C|m. \tag{3}$$

Where k is an arbitrary constant, we set to 2. To make things explicit, we use

$$n_{L,R} = n, \quad f_{L,R} = m, \quad n_C = m - s_C - \varepsilon, \quad f_C = f - s_C \tag{4}$$

as near and far for the corresponding cameras.

4.2 Mono-to-Stereo Mapping

At this point, all needed values are known to determine $P_{L,R,C}$ and $V_{L,R,C}$ and we can derive the mapping matrices

$$M_{C\to L,R} = P_{L,R}V_{L,R}V_C^{-1}P_C^{-1}, \quad M_{L,R\to C} = P_C V_C V_{L,R}^{-1}P_{L,R}^{-1} \tag{5}$$

to project arbitrary points from one clip space to another. We make use of $M_{C\to L,R}$ to project the near plane of C's clip space to the left and right clip spaces by setting $z = -1$ as the depth of a screen space aligned quad

$$Q = \left\{ \begin{array}{l} \hat{q}_{00} = (-1,-1,-1,+1)^T, \hat{q}_{10} = (+1,-1,-1,+1)^T, \\ \hat{q}_{01} = (+1,+1,-1,+1)^T, \hat{q}_{11} = (-1,+1,-1,+1)^T \end{array} \right\} \tag{6}$$

and yield

$$Q_{L,R} = M_{C\to L,R}Q. \tag{7}$$

Appropriate viewport dimensions w_C and h_C can then be calculated by

$$\begin{aligned} w_C &= \tfrac{1}{2} \min\left(x_{Lq_{10}} - x_{Lq_{00}}, \quad x_{Rq_{10}} - x_{Rq_{00}}\right) w_{L,R}, \\ h_C &= \tfrac{1}{2} \min\left(y_{Lq_{01}} - y_{Lq_{00}}, \quad y_{Rq_{01}} - y_{Rq_{00}}\right) h_{L,R}. \end{aligned} \tag{8}$$

(Note, the dehomogenization of the vertices q prior to this calculation.) Thus, resolution should match at the transition of the frustums of L/R and C as exact as possible. Using min ensures that pixel footprints of C are not smaller than those of L or R. Otherwise, more aliasing might be introduced due to undersampling during the projection of the monoscopic content to the stereo images.

If m should be featured as an arbitrary value it might be convenient to calculate the maximum dimensions W_C and H_C for the nearest m that should be selectable, since w_C and h_C vary depending on the chosen m, Using W_C and H_C for memory allocation of the third texture circumvents the need of any reallocation after initialization. Thus, the valid range of the texture coordinates uv_C for the access of image C is $\left[0, \frac{w_C}{W_C}\right] \times \left[0, \frac{h_C}{H_C}\right]$ instead of $[0,1]^2$.

As mentioned in Sect. 4.1, the aperture angles for the left and right camera frustum are hardware dependent [25] and reveal a by the manufacturer performed calibration step. For our HTC Vive e.g., we yield $l_L = -1.391937, r_L = 1.247409, t_L = -1.464287, b_L = 1.468819$ for the left eye and $l_R = -1.246557, r_R = 1.398447, t_R = -1.472458, b_R = 1.465505$ for the right eye (see Sect. 5 for details). The values are given as $\tan(\alpha)$, where α is an aperture angle from the z-axis to the respective frustum plane. The calibration is notable due to the fact that the aperture angles are, even besides of the usual skew, slightly asymmetric (compare e.g. l_L to r_R). The asymmetry causes minor differences in the size of pixel footprints at the mono plane (in spite of the adaption of viewport dimensions as described in Eq. 8). It is thus insufficient to perform a simple, pixelwise copy of a cropped, version of image C to realize the mapping, as this would highly increase the possibility of discontinuities along the mono plane. Hence,it is inevitable to allow for a subpixel-fine sampling when mapping to L and/or R. Bilinear texture filtering solves this problem, however at the price of slight blurring of the monoscopic image.

Furthermore, the bilinear filtering causes the need of some adaption if temporal anti-aliasing (see Sect. 4.5) is utilized. Though, it is highly recommended to apply some form of filtering during the reprojection. Using only nearest neighbor interpolation instead, leads to notable "snapping" of pixels along the lines where the offset exceeds a texel boundary.

In respect to pipeline design and further postprocessing step, we want to mention that after the assembly of a hybrid image pair the information of the origin of a pixel is lost (if not additionally tracked). In consequence, also the fragment depth is unusable from this point, except it was saved in a common range (e.g. world coordinates etc.) or transformed into a joint frustum during an intermediate step after rendering of the individual frusta. An alternative implementation using identical near- and far-planes together with hardware clipping planes for frustum division removes such restriction, but at the cost of decreased depth resolution.

4.3 Transition Artifacts

Without the overlap ε (see Fig. 2 left) we noted the occurrence of a cutting line along the mono plane which can be seen in Fig. 4. The probability of this occurrence depends on the depth difference of adjacent pixels at the mono plane which can be arbitrarily high. Hence, we linked ε to m (see Appendix) due to the pixel footprint which increases with distance and thus, a pixel can cover an even bigger depth difference. In theory, it may still happen that a rasterization ray exactly hits a point which neither is covered by the stereo nor mono frusta because of the perspective discrepancies in cases of very flat geometry, but by the linkage of m and ε it gets very unlikely.

Figure 4: Transition artifacts. Left: Cutting line. Right: Interpolation of view vectors $v_{L,R,C}$ within a user defined range.

In context of shading, we have to take care of slightly different view angles. While the transition is mostly seamless for diffusely lit pixels, there might be a noticeable seam through specular highlights and reflections. We propose a linear interpolation of viewing vectors to mitigate the artifact, see Fig. 4 right. The range where we interpolate is defined arbitrarily (we set ≈ 0.3 m). A conversion to quaternions was not necessary to achieve reasonable results, even though this operation complies with an interpolation of rotations.

4.4 Culling

Culling of invisible triangles and pixels is mandatory to achieve a speedup from hybrid stereo rendering. Efficient culling requires sufficiently fine grained object structures, which we assume in the following culling steps.

View Frustum Culling We use a single view frustum culling pass to simultaneously distribute scene objects to the mono- and stereoscopic cameras. Therefore, we cull against the union of all three frusta and the mono plane. Objects intersecting the mono plane need to be rendered thrice, which underlines the need of a fine grained model structure.

Merged Depth Map As we first render the left and right view L and R, we can perform additional culling for the far region. We merge the resulting depth maps and use this as a basis for culling objects and pixels when rendering the monoscopic view C. To ensure that an object beyond the mono plane cannot be seen from the left nor the right eye, we combine the left and right depth map to an image-space z pyramid [11]. One pixel of layer i holds the maximum depth value of an area of $2^i \times 2^i$ pixels in native resolution. We skip to write layer 0 and 1 to reduce the fill rate and unify the depth of left and right during the creation of layer 2.

Discarding Mono Fragments Based on our depth map, fragments can simply be discarded by sampling layer 2 and check if $z < z_m$, where z_m is the depth of the mono plane $m - \varepsilon$ in the clip space.

Occlusion Culling As discarding of monoscopic content only on fragment level might be wasteful, we use the remaining layers of the depth map to perform occlusion culling which comes at low additional fixed costs. The culling is straight forward: bounding boxes of the remaining objects after the frustum culling are projected to the clip space of camera C. In case the box is fully behind m, the size of the screen space bounding box is used to determine the depth layer to sample from and the depth values are compared.

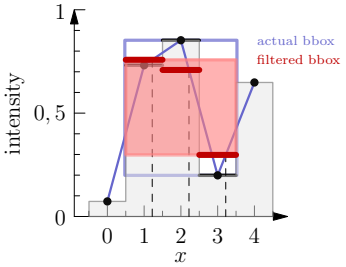
4.5 Anti-Aliasing

Proper anti-aliasing is mandatory, in particular for current HMDs, where the perceived footprint of one pixel usually exceeds that of conventional displays. In the following, we discuss the inclusion of Multi-sample Anti-Aliasing (MSAA) or Temporal Anti-Aliasing (TAA) into our pipeline.

MSAA [16] has the advantage of being fast and to be natively supported by the majority of hardware. Current implementations of MSAA require the rendering into a specialized framebuffer, which can hold multiple color and depth samples per pixel. The specialized framebuffer cannot be displayed directly and has to undergo a resolve into a conventional texture. This happens during a hardware accelerated blit which simply averages the colors for one pixel according to its depth samples [16]. Hence, it is rather inadequate in combination with hybrid stereo rendering. Using the conventional hardware resolve prior to the image assembly causes artifacts especially at the mono plane because of incomplete color and depth information. Whereas the use of the hardware resolve after the image assembly needs an expensive per sample mapping of the depth and color information. Additional clip planes and adapting the projection matrices (all with same near and far values) are needed to provide depth values in a common range. To circumvent the per sample writing accesses, a custom resolve is needed, too.

TAA is an efficient method to take care of sampling artifacts not only in spatial but also in the temporal domain. In concept, our implementation resembles Karis' [15] with some adaption to be applicable to our setup. TAA equals a temporal super sampling, where the positions of raserization rays are jittered across the pixel area and hence, vary every frame. We apply the same jitter for each camera, even though the pixel foot prints slightly differ due to asymmetries of our setup. In case of movement, though, the content covered by one pixel alters and is not temporally stable.

Common implementations of TAA, as also described by Karis, use color box clipping after the backprojection of a pixel. The clipping prevents an effect, known as "ghosting", by cutting the color history in case the new color differs too much from its history, e.g. in case of occlusions. The image interpolation applied during the projecting of the monoscopic contents into the stereo images influences the resulting bounding box, though. The effect is visualized in the inset image on the right. Since the color boxes tend to be smaller than usual (size of 0 in the worst case), aliasing is reintroduced especially at edges. One solution is to keep track of additional information, e.g. if a pixel in the assembled hybrid image pair has mono- or stereoscopic origin or calculate a scale parameter for the box based on the original pixel neighborhood. It would be also possible to write out the actual min and max of the adjacent HDR colors. But such approaches lead to extra overhead and need thought-out edge-treatments along the mono plane due to incomplete color information prior to the image assembly.



An alternative approach is to additionally scale the color box according to the current screen space velocities which are available anyway. By doing so, we significantly lowered the reintroduction of aliasing efficiently.

Besides of that, we also have to correct the screen space velocity of camera C v_C as it is computed in its original clip space and scale it respectively. Therefore, we calculate

$$v_{L,R} = \frac{1}{2} S v_C, \quad \text{with}$$
$$S = \begin{pmatrix} x_{L,Rq_{10}} - x_{L,Rq_{00}} & 0 \\ 0 & y_{L,Rq_{01}} - y_{L,Rq_{00}} \end{pmatrix} \tag{9}$$

similar to the already known Eq. 8. Note, that while filtering color during the reprojection is necessary for reasonable fitting along the transition, linear filtering should not be applied to the velocities.

5 Results

Our forward renderer, as depicted in Fig. 3, writes to six 16bit channels (RGBA & screen space velocity). During fragment processing, there are reading accesses to the shadow texture (3×3 percentage closer filtering) and the sky texture (skybox reflections), as well as to the respective diffuse and alpha textures of the material. We featured shadow, as it is known to be an important depth cue [4], in order to provide sufficient brightness and contrast conditions during our user study. We made use of `DrawElementsIndirectCommand` Buffers [12] to control the draw calls of the subdivided scene mesh. The frustum culling was performed on the GPU using a compute shader which only changed one field of the draw command struct to en- or disable the draw call for each mesh part. Hence, communication between CPU and GPU was minimized and had hardly measurable impact on the overall performance. Backface culling was enabled, too.

5.1 Performance

We evaluated the performance of our hybrid renderer with an own 3D scan of a terrain scene (Landscape in the quality assessment) and NVidia's Emerald Square Scene [14] using two different hardware setups. Setup A was made up of a NVidia GTX 1080Ti and a HTC Vive Pro. We rendered at the recommended resolution of 2352×2612 px per eye (return values of `GetRecommendedRenderTargetSize` [18]). Setup B represents a "low budget" configuration, made up of a NVidia GTX 970 and a HTC Vive, using the recommended resolution of 1344×1512 px [18]. The near plane distance n was set to 0.1 and the far plane distance f to 1000. The field of view parameters were set according to the return values of the OpenVR function `GetProjectionRaw` [25].

A breakdown of the time measurements including the single stages of our pipeline is presented in Fig. 5. We analyzed three variants of our pipeline with successively more optimization steps enabled. **No Opt.** indicates the rendering times without any depth map based optimizations only relying on the result of the frustum culling, **Discard** with discarding of fragments at the early beginning of the pixel shader stage of the third render pass based on the merged depth map, and **Occl. Cull** with additional occlusion culling. Frustum culling was enabled during all configurations (for the reference and **No Opt.**, too). Consequently, the reference amount of primitives rendered for each view varied.

The sampling distances of m cover those used for the user study in order to allow a trade-off analysis between performance gain and quality loss. Additionally, times were measured for $m = f = 1000$ providing a rough estimate of the minimally introduced overhead. The views were chosen to give an representative overview of possible time savings depending on the mono plane distance m. Therefore, the scene geometry was preferably uniformly distributed. We demanded not having too many occluders in the very near range to avoid skewed results in respect to depth based optimizations, as well. Simultaneously, all rendering times should preferably not exceed $\approx$ 11 ms (serving a 90 Hz framerate) to provide a realistic scenario with the use of HMDs.

The **Discard** variant of our pipeline was chosen for the in-depth analysis in Fig. 5. We identified the **Discard** variant as representative because it includes any additionally introduced overhead up to the depth map generation but is less scene dependent as the **Occ. Cull.** variant (with occlusion culling were far bigger time savings possible, though). We base this assumption on the observation that objects connected to the ground and the ground itself will rise in the visual field with increasing distance and thus, will make up a continually bigger proportion of the rendering.

In respect to performance, time savings were realistic for $m \leq$ 31 m (see e.g. both plots of Setup A and lower plot of Setup B of Fig. 5) for any of the three variants **No Opt.**, **Discard** or **Occl. Cull.**.

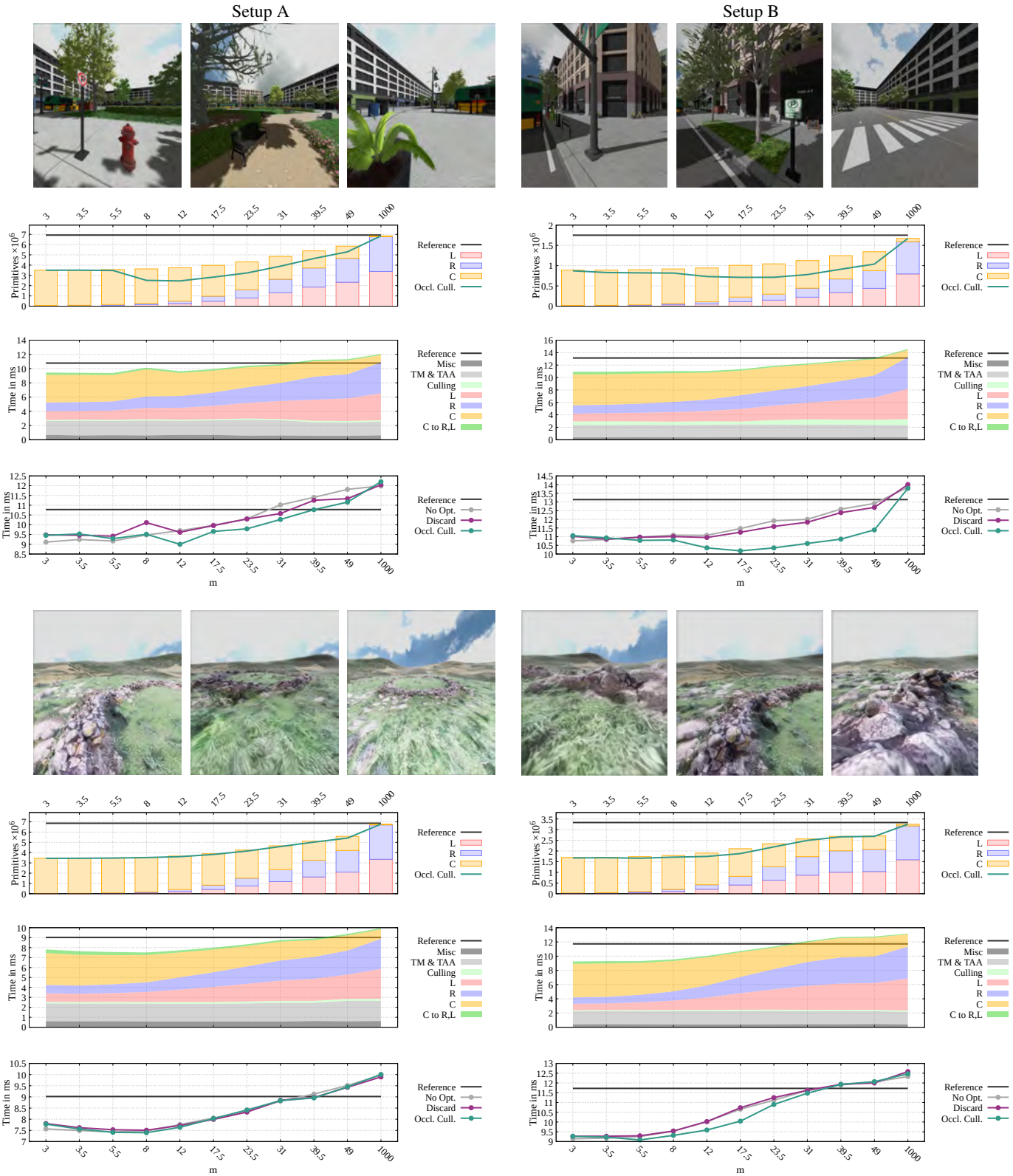


Figure 5: Averaged rendering times (lower two plots) and drawn primitives (upper plot) according to the distance of the mono plane m for Setup A (HTC Vive Pro & Nvidia GTX 1080Ti, left) and for Setup B (HTC Vive & Nvidia GTX 970, right). The views to measure the times are shown above the plots. **No Opt.**, **Discard** and **Occl. Cull.** are the three pipeline configuration variants with varying optimization steps. (**L**)eft, (**R**)ight and (**C**)enter indicate the camera specific amount of drawn primitives and rendering times into a HDR texture. **Misc** is mainly made up by the submission of the renderings to the HMD. **TM** is the tonemapping and **TAA** the temporal anti-aliasing performed in a single step. **Culling** includes the view frustum culling as well as the depth map generation as described in Sec. 4.4. **C to R,L** stands for the mapping of mono to stereo contents.
In the upper plot, the bars indicate drawn primitives only with frustum culling, whereas **Occl. Cull** is the amount of drawn primitives with additional occlusion culling. The cumulative plot in the middle disassembles the needed computation times with discarding of fragments and thus, includes the depth map generation but without occlusion culling. **Culling** covers the overhead needed for the workload reduction, namely frustum culling and the creation of the depth map (despite of the sampling of the depth map which is performed at the beginning of the fragment stage of the render pass for C).

Beyond this distance, the overhead of the additional renderpass, the mono-to-stereo mapping and the depth map generation usually outweighed any time savings. Of course, with **No Opt.** the best case occurs if no object is left in the stereo frusta, thus cutting down the amount of rendered primitives by half. For both, the **Discard** and the **Occl. Cull.**, holds that the biggest performance gains can be achieved if there is one fragment per texel rendered in the left and the right frustum. Consequently, the merged depth map is activated, and hence the workload for the third render pass is lowered, as well. This behavior is reflected e.g. in the upper plots of Setup B. For the range $m < 8$, the gain compared to the **No Opt.** variant is close to zero or even negative. For $m \geq 8$, the rendering times for **Discard** and the **Occl. Cull.** drop further, which also correlates with the decrease in the primitive count with applied **Occl. Cull.**. The LANDSCAPE scene hardly shows differences between the variants. This can be justified by the low depth complexity in the scene which makes it rather unsuitable for additional depth-based optimization approaches.

Overall, the numbers of Fig. 5 clearly show that the introduced overhead is very small and that a large fraction of the scene primitives is only rendered once even without depth based optimizations. By applying depth based optimizations additionally, rendering times dropped even further, especially in a range of $8 \leq m \leq 31$ which fits well considering the diminishing impact of binocular disparities as depth cue beyond 10 m.

5.2 Quality

In general, the achieved rendering quality of our system is on par with traditional stereoscopic rendering. Also the results of a questionnaire used in our user study (see Supplemental Material) indicate that the difference in perspective is the main difference between the renderings.

The inevitable kink in geometry from the slight change in perspective at the mono-plane becomes visible when layering a conventional and a hybrid rendering as done in Fig. 6. It appears along edges which are congruent up to the mono plane and then drift apart.

Using conventional image quality assessment methods, like the Structural Similarity Index (SSIM) [26], to compare each half of our hybrid image pair against the corresponding half of the conventional image pair individually, the measured discrepancy follows the formal error as expected. As depicted in Fig. 6, the difference is clearly dependent on the specified distance m. Even though, SSIM is suggestive of the introduced error, such metrics are inadequate to solely evaluate the coherence of stereoscopic image pairs; since they only take discrepancies on a single image basis into account but do not consider the image fusion performed by the HVS. Note that, for visualization purposes, all comparisons based on the SSIM were performed prior to any lens corrections, like the barrel distortion or the chromatic aberration correction. Therefore, the slight blurr caused by the mono-to-stereo mapping has hardly impact on the display which matches the findings of our questionnaire.

6 USER STUDY

Besides analyzing geometric error bounds and color differences, we were interested in evaluating the qualitative influence of the mono plane. Therefore we performed a user study to compare images resulted of the hybrid stereoscopic rendering and the common stereo rendering as reference mode. The study was intended to answer the following questions: Is there a loss of quality or comfort for the viewer? Is there a notable discrepancy between the rendering, and how does it vary with the distance of the mono plane. To explore these issues, we used three different scene types for visual inspection at ten levels of the distance to the mono plane.

6.1 Study Design

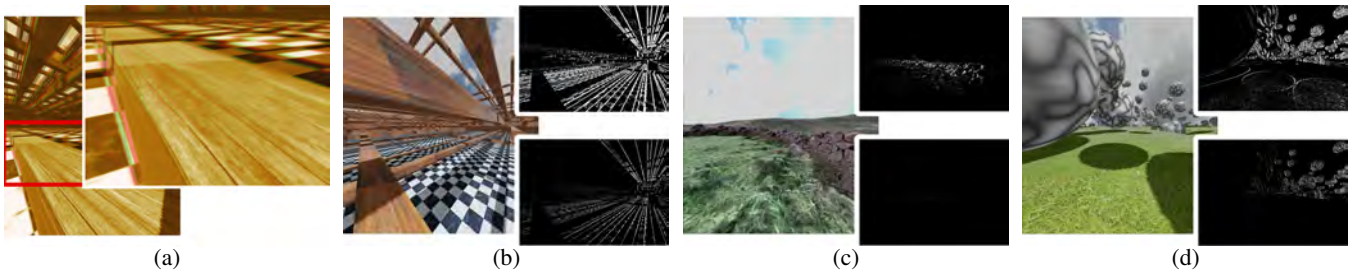
The user study included 47 subjects (16 females and 31 males, mean age 26.4) and was conducted as a within-subject study. The task was a simple decision task, which met the demand of the two-alternative forced choice (2AFC) technique. The two alternatives presented were the reference image (Stereo) and the hybrid stereoscopic image pair (Hybrid), both showed by an HTC Vive with maximum frame rate and same lighting conditions. The users were asked to make a forced choice between the presented images, i.e. choosing the preferred rendering. The images presented were three test scenes varying in their degree of realism and in the nature and arrangement of shapes: BEAMS, LANDSCAPE and SPHERES (see Fig. 6). BEAMS is a scene with many lines and edges. It should make visual flaws well visible, so it can be labeled as the worst scene regarding to our approach. SPHERES is the positive equivalent to BEAMS. There are no lines or edges but only spherical objects. LANDSCAPE, as natural scene, can be seen as a typical application scenario. All three scenes provide geometry which is uniformly distributed across space, and, thus, sufficiently meet our test range.

After an introduction session, the subjects viewed all three test scenes for ten distance levels, respectively, in both rendering modes. Thereby, the subject could manually switch between the modes as often as required and viewing the modes was not limited in time. The subjects never knew which rendering mode was presented. As soon as the participant was ready for decision making, he or she confirmed the preferred rendering mode and the next distance setup was displayed automatically. Whether the scene type or the distance or the rendering mode, all levels assigned to these independent variables were presented in a randomized order. Besides decision making, subjects had to fill in a self-metric questionnaire at the end of their session. This questionnaire was used for checking variances in qualitative matters between the options and the occurrence of any apparent side effects besides the known introduced geometric error.

6.2 Results

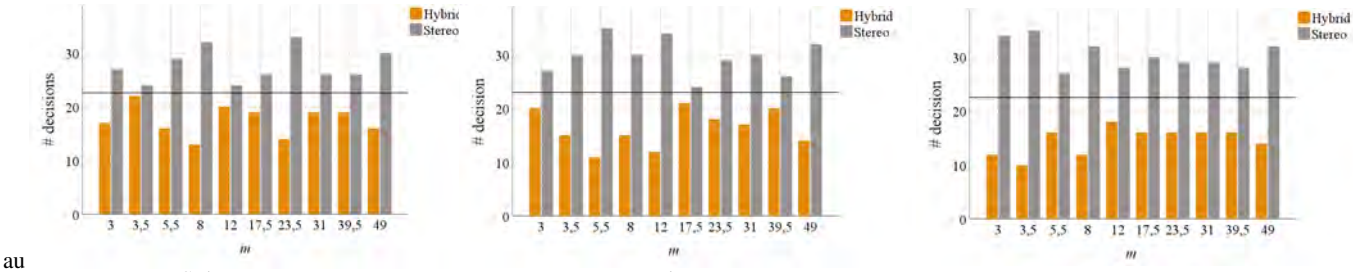
Overall, 47 participants had to make 10 decisions for three scenes, which results in 1410 decisions. Because sometimes the user did not viewed both rendering modes before making the decision, 48 decisions were declared as invalid. Thus, the number of observations used for statistical analysis amounted to 1362.

The results do not show any trend related to the distance m of the mono plane. Thereby hybrid-choices range from 29 % ($m = 8$) to 41 % ($m = 39.5$). Statistical tests also did not indicate any noteworthy correlations between the variable distance and others. Variables distance and decision have a weak correlation (phi = 0.07, $p > 0.05$) but it is not statistically significant. These results indicate that the detection of differences between the rendering modes was not only based on the projection error, which quickly decreases with growing m. One possible explanation would be that our hybrid rendering pipeline performs worse in terms of anti-aliasing, or that the general image quality is reduced, e.g. due to blurring. Answers to the questionnaire, however, indicate no visible difference regarding this. Instead, it seems that participants detected the difference between the rendering modes by the unavoidable kink of straight lines passing the mono plane as provided by the BEAMS scene. As soon as such anomaly has been found by a participant, it was easier to spot the difference. This is supported by visual inspection of the results for all three scenes over the distance levels (see Fig. 7). In general, participants preferred the fully stereoscopic rendering, but the preference strongly varies with the scene. While for the SPHERES the difference is moderate, for the BEAMS a strong preference for the fully stereoscopic rendering can be observed. Two-sample t-Test shows a difference in the means between the three scenes but the results are not statistically significant. Besides such differences between the scenes, there is also a variation in the choices regarding the order in which the participant saw the different scenes. Of the 47



(a) (b) (c) (d)

Figure 6: Disparity and scene views with local SSIM (visualized as $(1-SSIM)\times 10$). (a) Disparity between the reference (green) and the hybrid rendering (red) for the right image with $m = 1.6$ (a very close setting). It is apparent how the beam drifts to the left with increasing distance compared to the reference. Views of the scenes (b) BEAMS, (c) LANDSCAPE and (d) SPHERES with visualizations of their local SSIM ($w = 4$) using $m = 3$ (upper) and $m = 8$ (lower).



au

Spheres Landscape Beams

Figure 7: Decision for the different scenes for all m. The black line marks the 50% threshold. SPHERES: 175 decisions for Hybrid / 277 for Stereo ($\approx 38,72\%$) who adjudged the Hybrid mode as the better one. LANDSCAPE: 162 decisions for Hybrid / 297 for Stereo ($\approx 35,29\%$). BEAMS: 147 decisions for Hybrid /304 for Stereo ($\approx 32,59\%$).

subjects in total, 15 saw the scene LANDSCAPE at first, 16 SPHERES and 16 BEAMS. Participants whose first scene was SPHERES less often saw a difference in all scenes than participants who saw BEAMS or LANDSCAPE at the beginning. Regarding this, the data show that 40 % of the subjects with SPHERES at first, 37 % with LANDSCAPE and only 29 % with BEAMS at first chose our hybrid stereoscopic technique.

In summary, it can be said that the quality or comfort for the viewer varies with the type of scene or application and not with the distance of the mono plane. Thus, we cannot give a recommendation for the best distance for the mono plane, and one of our main questions must remain unanswered. As it turned out, there is no trend recognizable, for which distance it is advisable and for which distance it is not. Neither the decision time nor the switch-rate, have shown any interaction between the distance of the mono plane and the viewer's preference. One reason for this could be that once kink features have been discovered, the user was more triggered to detect such anomalies, even at greater distances.

The results of the questionnaire as well as a detailed description of the study and its results can be found in the Supplemental Material.

7 CONCLUSIONS AND FUTURE WORK

In this paper we showed that hybrid stereo rendering can result in significant performance gains. However, such performance gains require a reasonably grained subdivision of the scene, so that culling techniques perform well. Furthermore, there are always situations possible, where most of the geometry is in the near range, and thus has to be rendered twice, so the gain is not constant in all situations.

We did not examine transparent objects in our implementation, which would require more complicated merging of the images. Furthermore, we do not handle moving objects, which would require modifications to the culling stages and to temporal anti-aliasing.

The merging of the monoscopic view results in a linear filtering of the far region, which is unavoidable unless we could guarantee that the left and right frusta are symmetric. The result is that the far region is slightly blurred, but due to the following temporal anti-aliasing and the lens corrections this effect is hardly noticable. However, the effect should be taken into consideration for a future user study, and it should be checked if it does not negatively impact the perceived quality.

Finally, our user study gave us no hints what the best choice for the mono plane distance is. We can only give the advice to adjust m visually. It would be good to develop an approach to choose m adaptively, depending on currently seen objects.

The user study showed us that the applicability of our approach heavily depends on scene characteristics. However, we did not evaluate how big the impact is, and whether the approach is not usable for certain scenes or maybe tasks. We have the hypothesis that the inevitable kink in straight lines that pass the mono plane is noticeable, but we did not evaluate how big the impact of this effect on perceived quality is. We made the experience that the effect is particularly visible if a virtual laser beam is attached to a controller. It is possible that the approach has a negative impact on performing tasks in VR, that require interactions with objects beyond the mono plane, which should also be examined in future studies.

REFERENCES

[1] W.-Y. Chen, Y.-L. Chang, S.-F. Lin, L.-F. Ding, and L.-G. Chen. Efficient depth image based rendering with edge dependent depth filter and interpolation. In *2005 IEEE International Conference on Multimedia and Expo*, pp. 1314–1317. IEEE, 2005.

[2] J. E. Cutting and P. M. Vishton. *Perceiving layout and knowing distances: The interaction, relative potency, and contextual use of different information about depth.*, pp. 69–117. Perception of space and motion, 1995.

[3] R. V. Daniele Di Donato, Remi Palandri. High quality mobile vr with unreal engine and oculus. GDC 2017, 2017.

[4] P. Didyk, T. Ritschel, E. Eisemann, K. Myszkowski, and H.-P. Seidel. A perceptual model for disparity. In *ACM Transactions on Graphics (TOG)*, vol. 30, p. 96. ACM, 2011.

[5] DrashVR. Titans of space. http://titansofspacevr.com/tosclassic.php.

[6] T. Elbaum, M. Wagner, and A. Botzer. Cyclopean vs. dominant eye in gaze-interface-tracking. *Journal of Eye Movement Research*, 10(1), 2017.

[7] I. Epic Games. Monoscopic far field rendering. Unreal Engine (v4.16) Documentation. https://docs.unrealengine.com/en-us/Platforms/VR/MonoFarFieldRendering.

[8] I. Epic Games. Vr performance features. Unreal Engine (v4.17) Documentation. https://docs.unrealengine.com/en-us/Platforms/VR/VRPerformance.

[9] I. Epic Games. Release notes 4.15. Unreal Engine Documentation, February 2017. https://docs.unrealengine.com/en-US/Support/Builds/ReleaseNotes/4_15.

[10] C. Fehn. A 3d-tv approach using depth-image-based rendering (dibr). In *Proc. of VIIP*, vol. 3, 2003.

[11] N. Greene, M. Kass, and G. Miller. Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 231–238. ACM, 1993.

[12] K. Group. glmultidrawelementsindirect, 2014. https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glMultiDrawElementsIndirect.xhtml.

[13] T. Heath. Ort (near stereo, far mono). OculusSDK Samples, December 18 2014.

[14] N. Hull and N. Benty. Nvidia emerald square, open research content archive (orca), July 2017. http://developer.nvidia.com/orca/nvidia-emerald-square.

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

[16] D. Kirkland, B. Armstrong, M. Gold, J. Leech, and P. Womack. Multisampling, March 2002. https://www.khronos.org/registry/OpenGL/extensions/ARB/ARB_multisample.txt.

[17] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross. Nonlinear disparity mapping for stereoscopic 3d. In *ACM Transactions on Graphics (TOG)*, vol. 29, p. 75. ACM, 2010.

[18] J. Ludwig. Ivrsystem::getrecommendedrendertargetsize. OpenVR Wiki, April 2015. https://github.com/ValveSoftware/openvr/wiki/ IVRSystem::GetRecommendedRenderTargetSize.

[19] A. P. Mapp, H. Ono, and R. Barbeito. What does the dominant eye dominate? a brief and somewhat contentious review. *Perception & Psychophysics*, 65(2):310–317, 2003.

[20] B. C. McCann, M. M. Hayhoe, and W. S. Geisler. Contributions of monocular and binocular cues to distance discrimination in natural scenes. *Journal of vision*, 18(4):12–12, 2018.

[21] NVidia. Single pass stereo. NVidia Developer Documentation. https://developer.nvidia.com/vrworks/graphics/singlepassstereo.

[22] S. G. Remi Palandri. Hybrid mono rendering in ue4 and unity. Oculus Blog, 2016. https://developer.oculus.com/blog/hybrid-mono-rendering-in-ue4-and-unity/.

[23] A. Schollmeyer, S. Schneegans, S. Beck, A. Steed, and B. Froehlich. Efficient hybrid image warping for high frame-rate stereoscopic rendering. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1332–1341, April 2017. doi: 10.1109/TVCG.2017.2657078

[24] R. Srinivasiah. How to maximize ar and vr performance with advanced stereo rendering. Unity Blog, November 21 2017. https://blogs.unity3d.com/2017/11/21/how-to-maximize-ar-and-vr-performance-with-advanced-stereo-rendering/.

[25] R. Srinivasiah. Ivrsystem::getprojectionraw. OpenVR Wiki, April 2017. https://github.com/ValveSoftware/openvr/wiki/IVRSystem::GetProjectionRaw.

[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[27] M. Weier, T. Roth, E. Kruijff, A. Hinkenjann, A. Pérard-Gayot, P. Slusallek, and Y. Li. Foveated real-time ray tracing for head-mounted displays. In *Computer Graphics Forum*, vol. 35, pp. 289–298. Wiley Online Library, 2016.

[28] M. Weier, M. Stengel, T. Roth, P. Didyk, E. Eisemann, M. Eisemann, S. Grogorick, A. Hinkenjann, E. Kruijff, M. Magnor, et al. Perception-driven accelerated rendering. In *Computer Graphics Forum*, vol. 36, pp. 611–643. Wiley Online Library, 2017.