FAVR - Accelerating Direct Volume Rendering for Virtual RealitySystems

Andre Waschk and Jens Krüger *Member, IEEE*

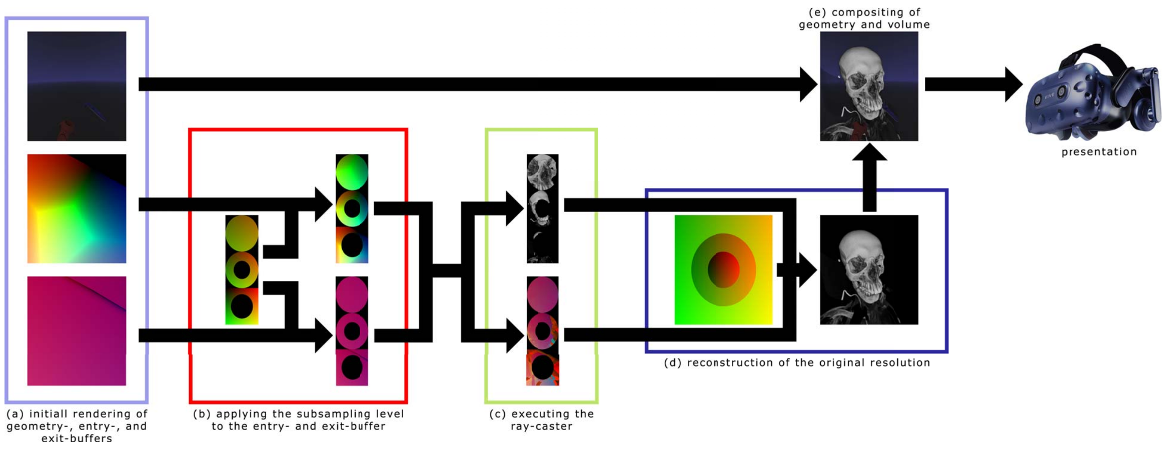
(e) compositing of geometry and volume
presentation
(a) initiall rendering of geometry-, entry-, and exit-buffers
(b) applying the subsampling level to the entry- and exit-buffer
(c) executing the ray-caster
(d) reconstruction of the original resolution

Fig. 1. We have implemented an additional subsampling (b) and reconstruction (d) stage to our rendering pipeline, which reduces the fragment load of our direct volume renderer and enables the usage within a VR context.

Abstract—In recent years, virtual reality (VR) has become readily available using robust and affordable head-mounted display (HMD) systems. Several VR-based scientific visualization solutions were proposed recently, but direct volume rendering (DVR) was considered only in a handful of VR-applications. We attribute this to the high computational demand of DVR and the limited rendering budget available for VR systems. For a heavily fragment-bound method such as DVR, it is challenging to achieve the very high update rates essential for VR. We propose an acceleration technique designed to take advantage of the specific characteristics of HMD systems. We utilize an adaptive rendering approach based on the lens distortion of HMDs and the visual perception of the human eye. Our implementation reduces the rendering cost of DVR while providing an experience indistinguishable to standard rendering techniques.

Index Terms—Volume Rendering, Application Motivated Visualization, Algorithms, Scalar Field Data

1 Introduction and Related Work

Volume visualization has established itself as an indispensable tool for the analysis of three-dimensional data. In particular, direct volume rendering (DVR) [1] is a popular method due to its ability to consider the original three-dimensional data. This approach requires powerful hardware due to the underlying computational complexity of its core visualization algorithm. State-of-the-art DVR utilizes parallel graphics processing units (GPU) to accelerate the essential evaluation of the ray-equation. These high-performance GPUs are mandatory to achieve performance sufficient for day-to-day use in most critical areas—the current standard for DVR a commodity workstation setups allowing users to visualize scanned or generated data-sets.

In recent years, multiple vendors have developed consumer solutions for VR, with Oculus [12] and HTC [5] being the most prominent competitors for consumer devices. The current standard in the consumer market are so-called head-mounted displays, which are self-contained virtual reality systems for immersive user experiences. As the name implies, these displays are worn on the head of the user and cover the entire visual field, completely excluding the real world.

Commonly, modern VR systems allow users to explore the spatial surroundings freely, only restricted by the physical boundaries of the real world. We assume that this freedom will motivate users to explore volumetric data in new ways.

However, this freedom in motion brings constraints for these systems in the form of the necessity of high update rates and low response times. Not achieving these mandatory constraints result in fatigue and nausea, also known as cybersickness [8]. Multiple different approaches are lately discussed how these cybersickness symptoms can be avoided [2,6, 9]. One of the most significant impacts seems to be the responsiveness of the system influenced by the refresh-rate of the HMD and latency of the tracking hardware. To avoid cybersickness symptoms, vendors equip HMDs with displays supporting refresh-rates of at least 90 Hz or more on high-end devices. These refresh-rates should also be seen as the lower boundary for any VR application.

Another critical key aspect of HMDs is the physical display resolution. While earlier generations of HMDs used displays with a native resolution of 2160 x 1200 pixels, the current state-of-the-art product line shifts to displays of at least 2880 x 1600 pixels. These high-resolution displays are necessary to present a sharp image close to the eye and reduce the appearance of screendoor-effects.

While those systems are primarily designed for the entertainment sector, we believe that an immersive visualization can be beneficial for the DVR context. The added stereoscopic projection enhances the spatial perception of the user, providing additional support for the recognition of critical structures. The added tracking capabilities of

• *Andre Waschk is with University of Duisburg-Essen. E-mail: andre.waschk@uni-due.de.*
• *Jens Krüger is with University of Duisburg-Essen. E-mail: jens.krueger@uni-due.de.*

HMDs further enhance the perception of the DVR visualization. Users can freely move around the visualized data and examine the data from every angle, making the experience much more natural.

The challenge for direct volume rendering, in combination with virtual reality, is the necessary high refresh-rate. VR demands consistent and high update-rates to avoid the appearance of cybersickness symptoms.

Several methods have been developed to accelerate DVR on modern graphics processing units [3, 7, 16] and those techniques improved the overall performance and applicability to the point where interactive exploration of large scale DVR data is standard on desktop systems. However, VR setups are a different story. For a VR application, we have to assume that the volumetric data can cover the entire screen when users dive into the data during exploration. As a result, the volume rendering system has to render the data to every single pixel on the screens. A common method to achieve high update rates are progressive rendering approaches, which render a quick preview during interaction and refine the image once the interaction stops. In VR, however, the interaction never stops, and those techniques are not directly applicable to head-tracked environments. Therefore, we propose a rendering method that reduces the fragment workload while immediately producing a high-quality result. To achieve this, we adjust the ray-casters sampling-rate to the non-uniform characteristics of the HMDs lenses and the human visual system.

This realization that straight forward acceleration techniques are insufficient for VR setups was also addressed by Scholl et al. [15]. They presented a solution for a direct volume rendering system on HMDs, too, and considered 90 Hz to be the minimum update-rate for suitable VR applications. Due to DVR's complexity, they had to limit the sampling rate during ray traversal to 150 samples along a single ray, limiting overall image quality. They also limited themselves to data-sets smaller than what we found to be commonly used in daily practice to maintain the necessary refresh rate.

Usher et al. [19] focused on a visualization tool for neuron tracing utilizing direct volume rendering in VR. While providing a useful tool in an immersive environment, they could only visualize small bricks of the original volumetric data in real-time to maintain sufficient update rates. Meng et al [11] presented another approach, for the acceleration of HMD rendering, which reduced the actual rendering resolution on the non-dominant eye.

To accelerate DVR for state-of-the-art consumer VR, we introduce Focus Adaptive Volume Rendering (FAVR). Our method efficiently reduces the number of fragments processed during rendering without perceiving loss in image quality. We adjust the rendering resolution in the peripheral vision of the human visual system, which is not further noticeable to the user [10] and utilize the fixed lens setup provided by HMDs. Similar to our approach, other solutions have been developed for desktop applications [4, 17, 18] and indicate a massive increase in rendering performance. Applying our technique, we can provide an interactive system well exceeding the mandatory update-rates for VR.

2 Focus Adaptive Volume Rendering

The evaluation of the ray equation is the costly part of DVR. It is heavily bound to the GPUs memory bandwidth due to the number of texture fetches mandatory for every single fragment. On a typical desktop setup, refresh rates of 30 Hz are mostly enough for interactive exploration of volumetric data, while HMD systems require a much higher refresh rate. Even low-end HMD devices exceed the resolution of commodity desktop monitors, especially considering the necessity to cover every single pixel of the HMD. As a possible solution to this problem, we present an acceleration method directly tailored to HMDs. Our approach reduces the number of fragments processed during DVR by considering the lens-to-pixel distribution of HMDs and the perception of sharpness in the peripheral vision.

2.1 Pipeline Description

We describe our general visualization pipeline as a deferred rendering approach. For the VR context, it is not sufficient to only visualize the volumetric data; in fact, correct compositing with polygonal models like

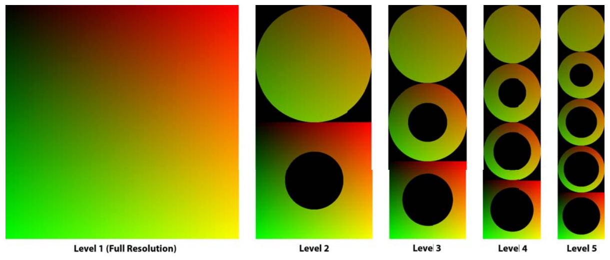

Fig. 2. For the fragment reduction, we precompute multiple lookup textures for our subsampling stage. These are used to generate entry- and exit-buffers with reduced information density in the peripheral vision.

the user's avatar is necessary. Our approach separates the visualization of the geometry and the volumetric data and only applies the here described acceleration method to the volumetric part of the pipeline, as seen in Figure 1.

For the visualization of the volumetric data, we use a ray-caster following the presented solution of Krueger et al. [7]. To accelerate the ray-traversal, we implemented early-ray-termination and empty-space-skipping. Our approach adds a subsampling stage before the computational intensive ray-traversal and a final reconstruction afterward. This subsampling stage is the fundamental fragment reduction part of our acceleration and allows us to achieve the necessary refresh-rate for HMD devices.

In a first step, GPU based ray-casters visualize the bounding geometry of the data in the form of entry and exit-buffers. Afterward, these buffers are used to traverse the actual data. In terms of HMDs, we have to consider framebuffers of the native display resolution for the best possible quality, which would heavily reduce the overall performance of the system. Thanks to the build-in lenses in HMDs, only the central part of the display can be perceived in its entire pixel density. We can harness this property and massively reduce the actual pixel coverage for the peripheral vision of the user without being noticeable.

Therefore we precompute lookup textures for the reduction and reconstruction of the full-resolution entry and exit-buffers. We describe the construction and usage of these buffers in the next Section. Our system can fully automatically adapt the subsampling level on the fly without any interaction of the user and maintains a stable refresh rate for the VR context.

2.2 Lookup Textures

As established, DVR's most computationally expensive part is the evaluation of the ray-equation for every single fragment covered by the volumes bounding geometry. Most techniques aim to accelerate the evaluation of this equation. While already applied to our renderer, these techniques are not enough for DVR on and head-mounted display. Considering the grown number of fragments on HMDs, compared to desktop applications, ray-traversal acceleration techniques are insufficient, and the actual number of fragments has to be reduced.

Therefore, our system precomputes multiple subsampling and reconstruction textures on startup. These subsampling (Figure 2) and reconstruction-textures (Figure 3) represent lookup tables mapping fragments from the incoming entry and exit-buffers of the bounding geometry to a target buffers for our ray-caster input. The number of precomputed textures, also called subsampling levels, can be specified by the user. During runtime, our system evaluates the computation time of prior frames and dynamically switches between different levels to maintain a refresh-rate above 90 Hz.

The subsampling textures in Figure 2 represent five levels of our acceleration method. On level 1, our system copies the original full-resolution buffers into the ray-caster target, reducing 0 fragments. From level 2 on, the system introduces the here seen circular structure to the pipeline. These structures are based upon the circular form found on HMD lenses.

In our system, we call these structures stages, and each texture itself a subsampling level. For each level l, we have the same amount of stages found in the texture. Level 1 consists of one stage, the direct lookup texture of the original input texture. Level 2 introduces the first two stages. The width of each subsampling texture is determined by

$$\frac{1}{l} \cdot w \tag{1}$$

where w represents the original width of the input texture. In our implementation, the height h of the texture stays the same.

Each consecutive stage s, starting from the top with 0, represents a scaled portion of the input buffer. The resolution of each stage is scaled by

$$1.0 - \frac{s}{l} \tag{2}$$

As the represented figures demonstrate is the overall number of fragments reduced to

$$w \cdot h \cdot \frac{1}{l} \tag{3}$$

for each subsampling level. The number of fragments evaluated during ray-traversal is reduced even further due to the circular structures and empty spaces in these textures. We evaluated the percentage of remaining fragments for each level and found out that subsampling level 2 already eliminates over 60% of the fragments and level 3, roughly 80%.

After evaluating the ray-traversal, using the subsampled entry and exit-buffers (see 1), the original sized image has to be recreated and presented to the user.

Therefore our system computes reconstruction textures 3, which map texels of the subsampled ray-caster image to a full-sized framebuffer. The resulting image of the ray-caster is partially rendered on a lower resolution than our final target image, making interpolation mandatory to fill the missing data. In our case, we are using bilinear interpolation, making the peripheral area of the image blurry. Also, to make sure that no apparent gaps are visible to the user and to smoothen the transition between stages, we added a small overlap between stages, which is used to blend between stages and remove the visible edge.

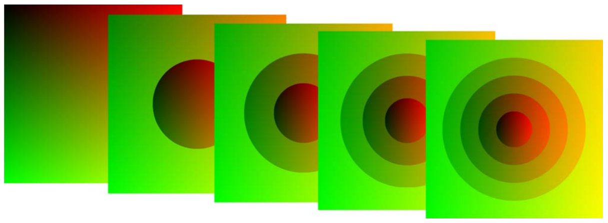

Fig. 3. We prepare lookup textures for the reconstruction of the final image. Each data point of these textures corresponds to the location of our source subsampling texture.

3 Results

To evaluate our acceleration method, we investigate three different aspects of our system. At first, we focus on the performance of our solution. Since we aim to achieve update rates of at least 90 Hz, we compare our solution to a traditional full-resolution rendering approach. In the second part, we investigate the loss in image quality introduced due to our subsampling and reconstruction stage. In the context of preliminary tests, we presented our approach to field experts and medical students to evaluate the perceived image quality and performance difference.

3.1 Performance

In this section, we focus on the evaluation of the performance increase using our acceleration method. We aimed to reach the refresh rate of 90 Hz, currently found in all high-end HMD systems. Also, We desired to design our system to be highly scalable for systems with higher refresh-rates, like the Valve Index, and bigger data-set than currently tested.

Our benchmark system was equipped with a Ryzen 7 2700X processor, 32 GB DDR4 3200 Mhz memory, and an Nvidia GTX 1080 graphics card to examine our rendering pipeline. As a VR system, we use the HTC Vive Pro with a display resolution of 2560 x 1440 pixels. To guarantee that each technique renders the same images, we recorded a fixed set of interactions filling a 3-minute benchmark sequence. The set of interactions consists of translation, rotation, and scaling tasks with additional changes in the transfer-function and real-time editing of the data-set. For the data-sets, we selected an MRI scan of the human torso with a resolution of 512 x 512 x 792 voxels, and a CT scan of the head with a resolution of 512 x 512 x 256 voxels as these are the most common resolutions of state-of-the-art scanners.

We examined four different rendering setups with three different sampling rates. For our baseline, we tested a ray-caster implementation with empty-space skipping and early-ray termination. In the actual system, our rendering pipeline automatically adjusts the subsampling level for each frame. We evaluate the last finished frame-time and adjust the subsampling level on demand. For each stage's performance analysis, we deactivated this feature to gain more detailed information about the performance increase for each level. We tested the subsampling level of two, four, and six, which we also analyzed in terms of image quality in Section 3.2. For the ray-traversal, we selected the sampling rates of 128, 512, and 1024 samples, where undersampling during the ray-traversal introduces significant visible artifacts but massively increased the rendering performance. Therefore it is mandatory to select a sampling rate reading each voxel of the data-set at least once.

Results 1 show that even low subsampling levels of our approach are sufficient to achieve update rates beyond the targeted 90 Hz where the baseline ray-caster could not maintain the necessary refresh rate on adequate sampling levels.

Overall we could achieve a significant speed-up in rendering performance compared to the other solution. For the MRI data-set and 512 samples, we gained an additional performance between 84.7% (level 2) and 119.9% (level 6) compared to the optimized ray-caster. Even the less demanding CT data-set showed a performance increase between 72.2% and 151,7%.

3.2 Image Quality

The reconstruction of the final frame undergoes a reconstruction stage that introduces artifacts and errors to the image that are not present in a full-resolution ray-caster used as a comparison to our technique. Since our approach is directly tailored towards HMDs and the build-in fixed lenses, it is impossible to convey the same perception in the form of images in this paper.

To evaluate our acceleration method and distinguish the difference between classical full-resolution rendering, we compute the Peak Signal-to-Noise Ratio (PSNR) of our solution. Therefore we rendered a fixed sequence of images with the same final target resolution and computed the PSNR for each of these images. We tested two, four, and six levels on three different data-sets. Figure 4 displays a set of example images of our evaluation. The zoomed-in portion accentuates the artifacts in outer regions of the image using our approach. Our PSNR analysis in Table 2 shows the average result for each data-set and subsampling level. While our solution is a lossy approach, we have to point out that the central focus area of the image keeps being lossless.

This analysis is focused on the objective result of our rendering technique but does not consider the actual user experience. We will talk in-depth about the user's perception in Subsection 3.3.

3.3 Preliminary Tests

The prior two Sections of this paper focused on the image quality and performance of our system. While we could establish that our

	MRI Torso			CT Head		
	128 samples	512 samples	1024 samples	128 samples	512 samples	1024 samples
[A] Baseline Ray-Caster	121.43	51.31	32.43	164.95	87.84	75.42
[B] FAVR LvL 2	138.33	94.79	51.46	239.38	151.64	101.19
[C] FAVR LvL 4	161.01	101.18	62.08	286.87	210.54	143.37
[D] FAVR LvL 6	193.11	112.30	74.81	300.00	221.12	153.12

Table 1. Benchmark results of our rendering method compared to a traditional ray-caster using empty-space skipping and early ray-termination *[A]*. We examined an MRI scan (512x512x792), and a CT scan (512x512x256) and compared different reduction levels *[B,C,D]*. The results represent the average frames per second (*fps*) achieved during a 3-minute benchmark.

	FAVR LvL 2	FAVR LvL 4	FAVR LvL 6
Torso	46.44 dB	39.97 dB	38.14 dB
Atery	49.66 dB	44.34 dB	40.04 dB
Head	47.16 dB	31.52 dB	39.99 dB

Table 2. We evaluated a sequence of images of the different data-sets and computed the average PSNR of these.

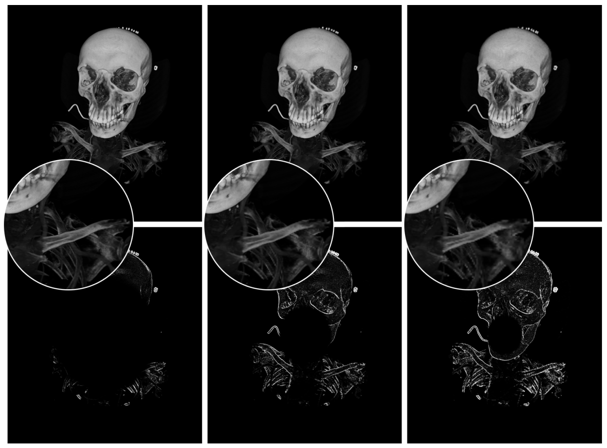

Fig. 4. We apply three different levels of our acceleration method to this data-set. From left to right, we demonstrate two, four, and six layers. The top row shows the final image, the bottom row the squared error compared to a full-frame ground truth image.

technique increases the system's performance, we also pointed out that our approach introduces artifacts in the peripheral vision. In this section, we focus on whether users could perceive these artifacts or not.

To validate our assumption that the artifacts in the peripheral vision are not noticeable by the user, we performed preliminary tests with 13 participants. For our tests, users explored multiple data-sets to their liking. We offer various interaction methods allowing them to translate, rotate, scale, and even edit the data-set in real-time. Especially the real-time editing and coloring of the data-set introduced an additional system load we had to compensate during run-time.

To evaluate whether users noticed any difference between our accelerated and a non-accelerated visualization, we started with a data-set of 128^3 voxels, which was still possible to visualize with at least 90 Hz on our baseline renderer. We switched between the full-resolution renderer and various subsampling levels during exploration. Afterward, we shifted to bigger data-sets of at least 512^3 voxels and tested whether the reduced refresh rate or the subsampling artifacts were noticeable to the users. The assumption that those artifacts are not perceivable by users is based on the fixed lens setup found in all currently available HMD systems and the human visual system. These lenses reduce the focusable area for users to a small portion in the center of the screen [13].

After exploring the 128^3 voxel data-sets, none of the users could notice a difference in image quality up to subsampling level 5; neither did they notice the adjustment at run-time. Starting from level 6, two of our participants could notice a reduced image quality in the center of the screen, which can be lead back to the fact that we now start to reduce parts within the focus area of the lens.

Once we established a soft threshold of subsampling level 5, we started exploring bigger data-sets of at least 512^3 voxels. As we outlined in Subsection 3.1, a ray-caster without our subsampling approach could not maintain refresh rates of 90 Hz. Again, we let the participants explore the volume, starting with a non-subsampled solution. All of the users noticed the low refresh rate, and some of them even had to stop the test to avoid cybersickness symptoms. For these participants, we had to continue our tests in the upcoming days. Once we enabled our dynamic subsampling solution, all users were immediately impressed by the system's fluidity.

After gathering all our participants' feedback, we could establish that low refresh rates are more noticeable to users and can lead to cybersickness symptoms. Our approach helps avoid those symptoms, and for the tested data-sets, we are far below our established noticeable subsampling threshold.

4 Conclusion

In our work, we presented a solution for the applicability of direct volume rendering on head-mounted displays. By introducing a subsampling and reconstruction stage to our rendering pipeline, we could reduce the fragment load for DVR and achieve refresh-rates far beyond the desired 90 Hz. Our solution is easy to adopt and highly scalable to a larger extend.

We tested the performance gains and scalability of our system compared to a non-subsampled solution by evaluating a benchmark scene on typical use-case data-sets. We discussed the image artifacts introduced due to rendering in a sub-native resolution and pointed out that these here clearly visible artifacts are not perceivable in a real-world application.

Furthermore, we evaluated our system in preliminary tests. We found out that none of the users could notice the difference in peripheral image quality in our fully automated and adaptive rendering environment. We again could validate that low refresh rates can lead to the rapid appearance of cybersickness symptoms, hence making our rendering approach a suitable solution of DVR in the VR context.

5 Future Work

In the future, we plan to combine our adaptive rendering system with state-of-the-art ray-guided volume rendering algorithms to tackle even larger data-sets interactively in VR.

Recently, several stand-alone VR systems have entered the market, and are considered by many the commodity VR hardware of the future. While providing significantly less performance compared to desktop computing hardware, those systems offer much more convenient access to VR. It will be interesting to explore the possibilities of direct volume rendering on these discrete devices. As those devices use similar hardware to modern smartphones, we expect a combination of volume rendering methods for mobile devices [14] with the techniques presented in this paper to enable direct volume rendering in VR even on these discrete VR headsets.

To further evaluate our technique, we are planning a full-scale user study to investigate the drawbacks and limitations of our approach and to quantify the effects of our subsampling approach. Furthermore, we consider multiple additions to our technique like a variable focus point, a more flexible arrangement and size for each resolution stage, and a more detailed explanation of our Level of Detail and editing system.

REFERENCES

[1] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pp. 65–74. ACM, New York, NY, USA, 1988. doi: 10.1145/54852.378484

[2] A. S. Fernandes and S. K. Feiner. Combating vr sickness through subtle dynamic field-of-view modification. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 201–210, March 2016. doi: 10.1109/3DUI.2016.7460053

[3] T. Fogal, A. Schiewe, and J. Krüger. An analysis of scalable gpu-based ray-guided volume rendering. *Proceedings. IEEE Symposium on Large-Scale Data Analysis and Visualization*, 2013:43–51, 10 2013. doi: 10.1109/LDAV.2013.6675157

[4] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Trans. Graph.*, 31(6), Nov. 2012. doi: 10.1145/2366145.2366183

[5] HTC. Vive discover virtual reality beyond imagination, 12 2019.

[6] A. Iskenderova, F. Weidner, and W. Broll. Drunk virtual reality gaming: Exploring the influence of alcohol on cybersickness. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '17, pp. 561–572. ACM, New York, NY, USA, 2017. doi: 10.1145/3116595.3116618

[7] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pp. 38–. IEEE Computer Society, Washington, DC, USA, 2003.

[8] J. J. LaViola, Jr. A discussion of cybersickness in virtual environments. *SIGCHI Bull.*, 32(1):47–56, Jan. 2000. doi: 10.1145/333329.333344

[9] J. J.-W. Lin, H. B.-L. Duh, H. Abi-Rached, D. E. Parker, and T. A. Furness. Effects of field of view on presence, enjoyment, memory, and simulator sickness in a virtual environment. *Proceedings IEEE Virtual Reality 2002*, pp. 164–171, 2002.

[10] L. C. Loschky, S. Szaffarczyk, C. Beugnet, M. E. Young, and M. Boucart. The contributions of central and peripheral vision to scene-gist recognition with a 180° visual field. *Journal of Vision*, 19(5):15–15, 05 2019. doi: 10.1167/19.5.15

[11] X. Meng, R. Du, and A. Varshney. Eye-dominance-guided foveated rendering. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1972–1980, 2020.

[12] Oculus. Oculus rift: Vr-headset, 12 2019.

[13] Oliver. The display resolution of head-mounted displays, revisited, 2018.

[14] A. Schiewe, M. Anstoots, and J. Krüger. State of the Art in Mobile Volume Rendering on iOS Devices. In E. Bertini, J. Kennedy, and E. Puppo, eds., *Eurographics Conference on Visualization (EuroVis) - Short Papers*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151139

[15] I. Scholl, S. Suder, and S. Schiffer. *Direct Volume Rendering in Virtual Reality*, pp. 297–302. 01 2018. doi: 10.1007/978-3-662-56537-7_79

[16] R. Sicat, J. Krüger, T. Möller, and M. Hadwiger. Sparse pdf volumes for consistent multi-resolution volume rendering. *IEEE transactions on visualization and computer graphics*, 20(12):2417–2426, 12 2014. doi: 10.1109/TVCG.2014.2346324

[17] M. Stengel, S. Grogorick, M. Eisemann, and M. Magnor. Adaptive image-space sampling for gaze-contingent real-time rendering. *Computer Graphics Forum*, 35(4):129–139, 2016. doi: 10.1111/cgf.12956

[18] Q. Sun, F.-C. Huang, J. Kim, L.-Y. Wei, D. Luebke, and A. Kaufman. Perceptually-guided foveation for light field displays. *ACM Trans. Graph.*, 36(6), Nov. 2017. doi: 10.1145/3130800.3130807

[19] W. Usher, P. Klacansky, F. Federer, P. Bremer, A. Knoll, J. Yarch, A. Angelucci, and V. Pascucci. A virtual reality visualization tool for neuron tracing. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):994–1003, Jan 2018. doi: 10.1109/TVCG.2017.2744079