Embracing Raycasting for Virtual Reality

Andre Waschk
Universität Duisburg-Essen
Lotharstraße 65
Germany, 47057 Duisburg
andre.waschk@uni-due.de

Jens Krüger
Universität Duisburg-Essen
Lotharstraße 65
Germany, 47057 Duisburg
jens.krüger@uni-due.de

ABSTRACT

This paper proposes an acceleration method for direct volume rendering (DVR). Our approach works like a wrapper or braces surrounding raycasting implementations and requires very few changes to the existing code. Visualization systems can significantly improve their rendering performance in virtual reality setups and make DVR feasible in these environments. The first step—the opening brace—modifies the initial ray construction by adaptively reducing the ray density, hence feeding fewer rays to the raycaster. The second brace step is a composting computation after the ray traversal that re-samples the raycasting results across the screen to reconstruct the final image. The rendering resolution is adapted during run-time to the specifications of the VR hardware and the performance of the renderer to guarantee stable and high refresh rates necessary to avoid severe cyber-sickness symptoms. The presented method utilizes gaze-dependent resolution levels tailored towards the human visual system (HVS) and hardware characteristics found in state-of-the-art head-mounted displays (HMDs). The resolution, and therefore the number of processed fragments during ray traversal, is reduced in the peripheral vision, delivering unnoticeable losses in image quality while providing a significant gain in rendering performance.

Keywords

Technique, Virtual Reality, Acceleration, Volume Rendering

1 INTRODUCTION

Due to the availability of cost-effective virtual reality hardware systems, Virtual Reality (VR) has, over the past years, found its way in many research fields of visualization [O'Leary et al., 2017, El Beheiry et al., 2019]. In particular, domains like astrophysics [Vogt and Shingles, 2013, Baracaglia and Vogt, 2020, Davelaar et al., 2018], engineering [Abulrub et al., 2011, Wolfartsberger, 2019, Wang et al., 2018], archeology [Novotny et al., 2019, Bruno et al., 2010] and also medical science [Huang et al., 2018, Chan et al., 2013, Chheang et al., 2021] have been of greater interest. Most of the publications mentioned suggest, that VR has the potential to improve the discovery and decision-making process in these domains as they benefit from the immersive, interactive, stereoscopic reproduction of complex spatial structures.

Many of the datasets in the aforementioned domains are not just surface structures but rather complex 3D volumes. A well-established approach to visualize volumetric data is direct volume rendering (DVR) [Drebin et al., 1988], which provides deep insight into the volume's data and structure. In the last few decades, significant advancements have been made to bring high-performance, high-quality direct volume rendering to commodity desktop systems [Krüger and Westermann, 2003, Fogal and Krüger, 2010, Meyer-Spradow et al., 2009]. Today, ray-guided volume raycasting systems have been widely adopted on traditional desktop systems with one or multiple monoscopic monitors. For virtual reality setups, however, raycasting is generally considered to be too computationally expensive.

Most affordable VR systems use head-mounted displays (HMD) and spatial tracking hardware to provide an immersive experience and offer spatial motion to the user. This freedom in motion and the usage of HMDs with the close proximity of the displays to the users' eyes come with significant constraints to the VR software. The users' near-continuous motion demands high refresh rates of the display and low latencies of the system. Even relatively short delays, which would be hard to notice on desktop systems, quickly increase the risk of fatigue and nausea, also known as cybersickness symptoms [LaViola, 2000]. Therefore, currently available VR systems utilize HMDs with update rates of at least 90 Hz with a tendency to 120 Hz or more.

In addition to high refresh rates, the close proximity of the display to the user's eyes also demands high resolutions. Present HMDs feature resolutions around 2880 x 1600 pixels with a trend to increase the resolution further to avoid the screendoor-effect [Anthes et al., 2016] and provide an ever-sharper image to the users. For algorithms that are performance bound by the number of fragments processed, the increased number of pixels poses a challenge. A DVR raycaster is such an approach.

In the last few decades, multiple different techniques [Levoy, 1990, Boada et al., 2001, LaMar et al., 2000, Zimmermann et al., 2000] have been developed to accelerate the computation of every individual ray. Early ray-termination, the intelligent choice of the correct transfer function, and progressive rendering [Fogal et al., 2013] have made it possible to visualize large datasets with refresh rates, which can be described as interactive but not high enough for VR usage. A well-established approach to scale rendering speeds is progressive rendering. During the interaction, the rendering quality is reduced by limiting costly computations, such as the number of rays or the sampling rate. When the viewport is fixed, e.g., mouse interaction seizes, the image is progressively refined. In a VR system, however, such a steady state is never really achieved as the camera is constantly adjusted to the tracking system's output.

The method presented in this paper is most closely related to the FAVR approach by Waschk and Krüger [Waschk and Krüger, 2020], i.e., that many of the pixels of the frame-buffer that are sent to the VR systems do not contribute to the perception of the scene. Although the nonuniform resolution distribution of the human-visual-system is well known [Resnikoff, 1989, Valois, 2000], and the use of this phenomenon in computer graphics and visualization is known as foveated imaging [Reder, 1973, Duchowski and Çöltekin, 2007, Murphy et al., 2009], FAVR considers not only human perception but also the nonuniform lens distortion of the HMDs (see Fig. 1). This research, in turn, builds on several works on gaze-dependent rendering [Bektaş et al., 2019, Reingold et al., 2003, Duchowski et al., 2005]. Even modern VR systems, equipped with retrofitted eye-tracking hardware, were considered by Vincent et al. [Vincent and Brannan, 2017] and Albert et al. [Albert et al., 2017]. With the recent developments in consumer VR systems, scientific visualization in VR environments has become of broader interest and has attracted many researchers' attention [Scholl et al., 2018, Usher et al., 2018, Chheang et al., 2021].

The improvements of this work over the FAVR-system can be summarized as follows:

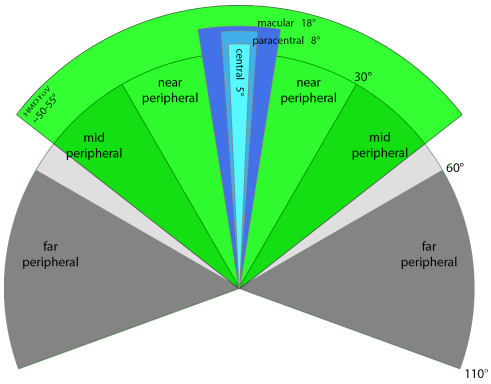

Figure 1: The human visual system is subdivided into numerous fields. Starting from the central vision, the perceived sharpness and the trichromatic perception are reduced the further the angle increases. Notice that modern HMDs cover only the central half of the visual field.

- simpler and more flexible ray-generation implementation
- load-based dynamic layer adjustment
- system scalability to lower end hardware

2 METHOD

To keep our acceleration method independent from the actual raycaster implementation, our method is split into two steps that are injected into existing pipelines (see Figure 2). Conceptually, our approach generates ray-entry and ray-direction information and stores those in texture maps (see Figure 2 (top-left)) to be used as input to the subsequent ray traversal (see Figure 2 (center)). The result of the ray traversal is handed back to our algorithm for final compositing (see Figure 2 (bottom)). Hence, the approach can be considered a bracket around existing ray traversal methods, which allows the combination of practically any ray traversal mechanism with our method, including recent ray-guided volume rendering systems.

We keep the visualization of the bounding geometry as our first step but add a subsampling stage before the ray evaluation. The subsampling-stage takes the entry and exit buffers as input and generates new buffers based on specific hardware characteristics of HMDs. This stage's resulting buffers still represent the bounding geometry but use multiple resolution levels to cover fewer fragments in total (see Fig. 3). The actual implementation and features of this stage are covered in Section 2. The images received from the ray traversal stage are now based on the subsampled input buffers and need to undergo a reconstruction method to generate the final output. To recreate a representation of the original image layout, we added a reconstruction stage to the pipeline, taking the resulting volume visualization and creating

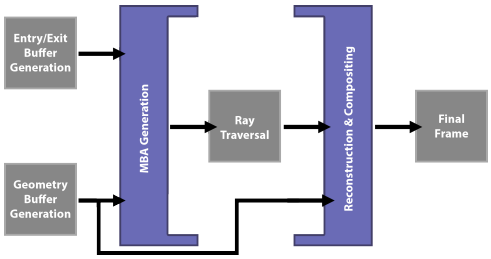

Figure 2: The processing pipeline with the addition of the multiresolution atlas buffer (MAB) generation as preprocess and the reconstruction of the full-resolution image after the ray traversal.

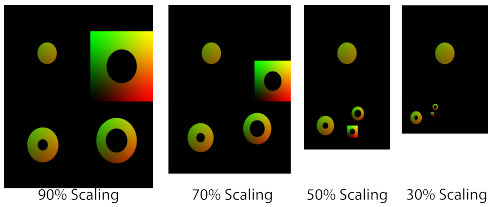

Figure 3: The multiresolution atlas buffer supports arbitrary resolution scaling values between subsequent levels. The resolution levels are ordered to minimize the overall framebuffer size.

a full-resolution image based on our pipeline parameters. Finally, in each frame, we evaluate the system's frame-timings and adjust rendering parameters for the next subsampling stage to maintain a consistent refresh rate (see Section 3).

2.1 Mutliresolution Atlas Buffer Construction

To reduce the number of primary rays that have to be traversed by the raycaster, the system adapts the spatial ray frequency to the decreased spatial perception in the peripheral region of the user's field of view. While foveated rendering systems also follow this approach, by utilizing high-speed eye-tracking hardware to locate the current gaze point, in the case of HMD setups, we do not need to consider the rapid eye movement of the users. Lenses built into HMDs are fixed and provide only a slight angle with the highest projected resolution.

The system uses discrete levels with progressively lower resolutions for specific portions of the image to reduce the number of rays. This layered approach is conceptually similar to MIP-Maps (see Figure 3). Compared to Mip-Maps, however, the system should support varying ratios between levels that are not necessarily a power of two, allowing us to further reduce the noticeable difference between levels.

To realize such a pipeline, our system has to construct the set of levels, forward the nonuniformly generated rays to the traversal, collect the results from the raycaster, and finally generate a consistent output image for display in the HMD.

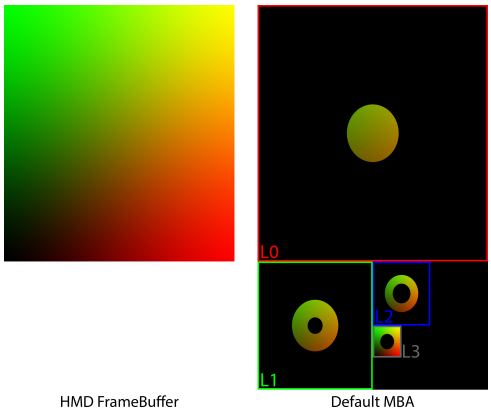

Figure 4: On the left: Screen-space framebuffer as it is used for the HMD rendering. Right: The default MAB setup of the system. The different resolution levels are color coded. The black areas in each level are discarded before the raytraversal.

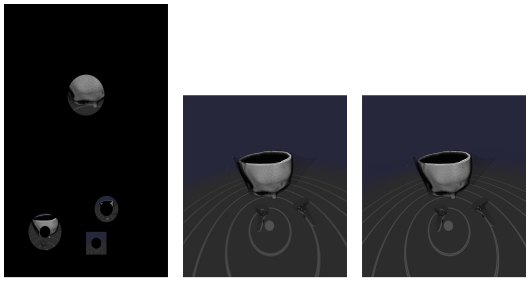

Figure 5: From the left to the right: The result of the raycaster rendered into the MAB. The full-resolution reconstruction that is submitted to the HMD. A native-resolution result as a comparison.

3 IMPLEMENTATION

To follow our approach, it is crucial to first understand the fundamental mapping between the HMD's frame buffer (s. Figure 4 left) and our multiresolution atlas buffer (MAB) (s. Figure 4 right).

The MAB is partitioned into multiple regions, where each region corresponds to the entire frame buffer at a specific resolution. This concept is similar to the levels of a MIP-map texture; however, in our approach, the difference between the various resolutions is not necessarily a fixed 4:1 ratio, which allows us to vary the resolution of each level independently.

A bidirectional mapping between the two spaces is mandatory to perform the two bracket steps of our approach, i.e., the computation of multiresolution ray coordinates for the raycaster and the compositing of the multiple resolutions into a single consistent framebuffer image. This function assigns to every MAB fragment a unique position in the frame buffer and a blending factor. This factor determines what resolutions will be mapped to this frame buffer in the compositing stage. The inverse of the function assigns to each framebuffer-

fragment several MAB fragments and blending factors for final compositing.

To explain the process, we look at the necessary steps in reverse order. After the ray traversal is complete, the compositing is initiated by rendering a full-screen-primitive that covers the entire output framebuffer. For each generated fragment, the system has to perform a mapping from the frame buffer to the MAB. This mapping is computed using two functions. The first function (f_a) takes as input the distance to the center of gaze, based on the screen-space coordinates of the fragment, and outputs a set of blending factors for the MAB. The second function (f_b) utilizes the screen-space coordinates of the fragment and outputs a set of corresponding coordinates in the MAB. Both functions have to be evaluated to perform the final compositing. For each nonzero-entry in the blending-factor set (f_a), a lookup into the MAB is carried out (f_b) to obtain the stored color values. These values are blended together into a single color for display.

Before this compositing step, the raycasting is initiated by rendering another full-screen primitive. This primitive, however, covers the entire MAB. Hence, the rasterizer generates all fragments for all resolutions of the frame buffer. To select only those MAB fragments that contribute to the compositing, we compute the inverse of the function mentioned above and determine the blend factor for a given MAB fragment. If the value is zero, the fragment is discarded.

To perform the inverse mapping, we use a third function (f_c) that takes as input the MAB coordinates and outputs two quantities, first, the resolution level of this fragment, and second the corresponding position in the frame buffer. The framebuffer position is used to determine the distance value for function f_a, where the resolution level is needed to select the correct blending factor returned by f_a.

In our current implementation, the function f_a is realized as a 1D lookup texture similar to a transfer functions used in direct volume rendering (Figure 6). The individual resolution levels of the MAB are mapped to the four color channels of the texture. The individual value of each channel describes the blending factor for the corresponding resolution level. To deliver unnoticeable blending, a smooth-step function is added between adjacent resolution levels (Figure 6).

4 FRAME RATE ADAPTION

The frame rate of a DVR system depends on a large variety of parameters. During run-time, even small parametric changes can significantly impact the performance, i.e., changing the transfer function used to sample the data or modifying the view frustum. A frame rate below a certain threshold is not desirable for virtual environments, and even small but abrupt dips in

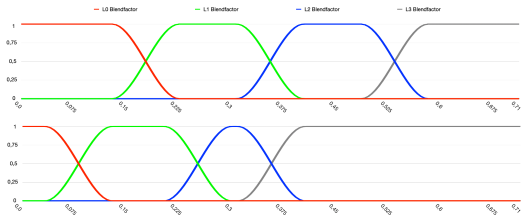

Figure 6: Two different resolution-distribution functions that are integrated into the system. Each line represents a specific resolution level of the MAB (Red = L0, Green = L1, Blue = L3, gray = L4). The Y-Value gives the opacity value for a fragment at the computed distance from the gaze. Values equal to 0 get discarded.

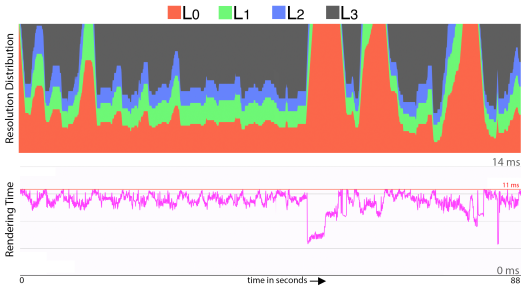

Figure 7: Adjustment of the distribution function is made during run-time. The sequence displayed is the result of our benchmark. The top displays the sizes of the different resolution levels and anchor points. The bottom displays the corresponding rendering time.

the rendering performance can lead to nausea and other cybersickness symptoms.

Parameters such as the camera position and viewing direction are especially variable when in a VR setup. The user's real-time tracking that enables immersive interaction is one of the major benefits of VR but it can lead to severe fluctuations in overall system performance.

To counter these effects, it is necessary to adjust the rendering system on the fly to maintain a consistently high update rate. The rendering method presented here provides a straightforward adjustment system to guarantee refresh rates above a set threshold. The system tracks the rendering performance and adjusts the resolution levels' distribution on the fly.

The function f_a (Section 2.1), implemented as a 1D-texture, is used in the opening brace as a fast lookup function to determine the corresponding resolution level of a texel and whether or not it has to be traversed by the raycaster. By modifying the texture, the different sizes of each resolution level can be adjusted.

To decide how the resolution levels should be distributed, the system tracks a sliding window of past n frames. First, the previous frame's rendering time is compared against two thresholds, a lower threshold of a minimum acceptable refresh rate and an upper threshold of a sufficiently high refresh rate.

If the performance is below the threshold, we shift the resolution distribution to cause fewer rays to be emitted. If the refresh rate is above the higher threshold, the system increases the number of rays. To compute the increment or decrements in rays, the system considers the gradient of the window of n-frames.

By evaluating the change in rendering time over the last n-frames, the system determines which anchor point between two resolution levels should be moved and how far it should be shifted in the corresponding direction. Minor variations lead to adjustments in the higher resolution levels, only slightly altering the resolution distribution, where high deviations primarily shift the lower resolutions to compensate in favor of the rendering performance (see Figure 7).

5 PERFORMANCE ANALYSIS

To test the scalability of our method, we evaluated our system on two hardware setups. We equipped a desktop computer with a Nvidia RTX 2070 as our graphics processing unit. As a second setup, we tested a laptop with a Nvidia RTX 2060 with a power limit of 80 watts to verify that our system is also scalable to a rendering setup. For our tests, we used the HTC Vive Pro as a VR system with a total display resolution of 2560 x 1440 pixels and a display refresh rate of 90 Hz.

We recorded a fixed set of interactions in the form of rotations, translations, scaling, and modifications in the transfer function to benchmark the system. To further extend our testing, we decided to test three rendering modes. The first mode, the baseline, is a full-resolution raycaster implementation utilizing empty-space skipping and early-ray termination. This mode represents a rendering setup without the opening and closing braces presented in this paper. The next mode tested is the default resolution level distribution (Figure 6) without any automatic adjustments and presents a uniform distribution of the resolutions across all levels. Finally, we benchmarked our method with the addition of automatic resolution adoption. We evaluated the minimum and average refresh rate on both hardware setups for all testes modes.

We selected two datasets for our benchmark common for state-of-the-art scanners, a CT scan with a resolution of 512 x 512 x 404 voxels and an MRI scan with a resolution of 512 x 512 x 392 voxels. To avoid undersampling the volume, we selected a sampling rate of 512 samples.

The results of our benchmarks, presented in table 1 show how our approach can increase DVR performance on HMD setups. The baseline tests on our desktop and mobile/laptop rendering setup could not maintain refresh rates even close to 90 Hz with an average frame rate of 65 fps on the RTX 2070 and 50 fps on the mobile RTX 2060 across both datasets. Using the default MAB setup without the addition of the dynamic resolution adjustments, the system could maintain an average refresh rates above 90 Hz for both datasets on the RTX 2070, but only on one of them for the RTX 2060. With the additional dynamic resolution distribution, the system could maintain the overall refresh rate above the threshold of 90 Hz on all devices

		RTX 2070		RTX 2060m	
		1%	avg	1%	avg
MRI	baseline	23.7	51.1	18.5	37.3
	(1) MBA	78.8	105.2	65.1	84.9
	(2) MBA	**90.3**	**108.2**	**90.1**	**91.2**
CT	baseline	37.6	79.0	26.8	62.6
	(1) MBA	80.8	132.9	77.4	97.1
	(2) MBA	**92.9**	**141.6**	**90.9**	**98.8**

Table 1: The table displays the rendering system's average performance on two datasets and two hardware setups. In addition to the average frame rate, we also display the 1% lowest frame rates. We compare a baseline ray-caster with our default (1) MBA setup and our dynamic (2) MBA system.

6 PRELIMINARY TESTS

The previous chapter 4 focused on the performance increment achieved by our acceleration approach. To validate our results, we performed preliminary tests. The study took 20 minutes on average and was conducted in our $16m^2$ lab using the same hardware setup as mentioned above. We recruited 17 participants, all of whom had previous experience with the exploration of volumetric datasets. In our test setup, participants could explore various datasets support by interaction methods such as translation, scaling, rotation, or transfer-function editing.

Our preliminary tests address two aspects of our system, first, is the loss in image quality noticeable to the users, and second, are dips below the rendering threshold more noticeable to the user than changes in the image quality.

To test these two aspects of our acceleration approach, we first let users explore datasets in a controlled environment in which we could verify that the default MAB could maintain 90 Hz most of the time. During exploration, we altered the resolution distribution, further reducing the image quality in peripheral vision. None of the participants noticed any change in overall image quality during exploration, supporting our assumption that the rendering resolution can be reduced in the peripheral vision in HMD scenarios.

To test the second aspect of our approach, we put more strain on the rendering system. We increased the size of the volumetric data and the sampling rate to stress the ray traversal further. Although the default MAB without additional resolution adaption could maintain an av-

erage refresh rate above 90 Hz, on some occasions and transfer-function setups, the refresh rate could dip to $\tilde{6}0$ Hz. With the dynamic resolution adaption added to the rendering pipeline, the refresh rate could be caped above 90 Hz at all times. At this point, we randomized the starting condition for each participant. Either the participants started with a fixed-resolution distribution including dips into low refresh rates, or they started with an adaptive resolution distribution but consistent refresh rate. After 1 minute of free exploration, the participant switched to the other mode. After collecting the feedback from each participant, none of them noticed that our system was changing the overall image quality on the fly, but all of them noticed that during the fixed MAB setup, the system was feeling less responsive, and two of our participants experienced nausea during the test.

7 CONCLUSION AND FUTURE WORK

In this paper, we presented a fast and applicable acceleration method for direct volume rendering in the VR context. Our system can provide an interactive exploration of volumetric datasets at 90 Hz or more by reducing the overall number of fragments processed during the computationally expensive ray traversal. Our system's scalability also provides an excellent opportunity to reduce the hardware requirement for volume visualization on hmd setups.

We conducted preliminary tests to assess the assumption that the loss in image quality in peripheral vision is not noticeable in the VR context. In our tests, we focused on the difference between consistent image quality and constant refresh rate. Our results showed that users did not notice changes in the image quality due to changing resolution zones, whereas rapid drops in the refresh rate could lead to nausea.

Thanks to our current implementation, the system can visualize the most common data-set sizes in real time with high refresh rates on a large scale of hardware setups. For the future, we plan to combine our approach with modern state-of-the-art ray-guided rendering systems to visualize even larger datasets interactively in VR.

8 REFERENCES

[Abulrub et al., 2011] Abulrub, A. G., Attridge, A. N., and Williams, M. A. (2011). Virtual reality in engineering education: The future of creative learning. In *2011 IEEE Global Engineering Education Conference (EDUCON)*, pages 751–757.

[Albert et al., 2017] Albert, R., Patney, A., Luebke, D., and Kim, J. (2017). Latency requirements for foveated rendering in virtual reality. *ACM Trans. Appl. Percept.*, 14(4).

[Anthes et al., 2016] Anthes, C., GarcÃa-HernÃ¡ndez, R. J., Wiedemann, M., and KranzlmÃ¼ller, D. (2016). State of the art of virtual reality technology. In *2016 IEEE Aerospace Conference*, pages 1–19.

[Baracaglia and Vogt, 2020] Baracaglia, E. and Vogt, F. (2020). E0102-vr: Exploring the scientific potential of virtual reality for observational astrophysics. *Astronomy and Computing*, 30:100352.

[Bektaş et al., 2019] Bektaş, K., Çöltekin, A., Krüger, J., Duchowski, A. T., and Fabrikant, S. I. (2019). Geogcd: Improved visual search via gaze-contingent display. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, ETRA '19, New York, NY, USA. Association for Computing Machinery.

[Boada et al., 2001] Boada, I., Navazo, I., and Scopigno, R. (2001). Multiresolution volume visualization with a texture-based octree. *The Visual Computer*, 17(3):185–197.

[Bruno et al., 2010] Bruno, F., Bruno, S., De Sensi, G., Luchi, M.-L., Mancuso, S., and Muzzupappa, M. (2010). From 3d reconstruction to virtual reality: A complete methodology for digital archaeological exhibition. *Journal of Cultural Heritage*, 11(1):42–49.

[Chan et al., 2013] Chan, S., Conti, F., Salisbury, K., and Blevins, N. H. (2013). Virtual Reality Simulation in Neurosurgery: Technologies and Evolution. *Neurosurgery*, 72(suppl1):A154–A164.

[Chheang et al., 2021] Chheang, V., Apilla, V., Saalfeld, P., Boedecker, C., Huber, T., Huettl, F., Lang, H., Preim, B., and Hansen, C. (2021). Collaborative VR for Liver Surgery Planning using Wearable Data Gloves: An Interactive Demonstration. In *Proc. of IEEE Conference on Virtual Reality (IEEE VR)*, Lisbon, Portugal.

[Davelaar et al., 2018] Davelaar, J., Bronzwaer, T., Kok, D., Younsi, Z., Mościbrodzka, M., and Falcke, H. (2018). Observing supermassive black holes in virtual reality. *Computational Astrophysics and Cosmology*, 5(1):1.

[Drebin et al., 1988] Drebin, R., Carpenter, L., and Hanrahan, P. (1988). Volume rendering. volume 22, pages 65–74.

[Duchowski et al., 2005] Duchowski, A., Cournia, N., and Murphy, H. (2005). Gaze-contingent displays: A review. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 7:621–34.

[Duchowski and Çöltekin, 2007] Duchowski, A. T. and Çöltekin, A. (2007). Foveated gaze-contingent displays for peripheral lod management, 3d visualization, and stereo imaging. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(4).

[El Beheiry et al., 2019] El Beheiry, M., Doutreligne, S., Caporal, C., Ostertag, C., Dahan, M., and Masson, J.-B. (2019). Virtual reality: Beyond visualization. *Journal of Molecular Biology*, 431(7):1315–1321.

[Fogal and Krüger, 2010] Fogal, T. and Krüger, J. (2010). Tuvok, an Architecture for Large Scale Volume Rendering. In Koch, R., Kolb, A., and Rezk-Salama, C., editors, *Vision, Modeling, and Visualization (2010)*. The Eurographics Association.

[Fogal et al., 2013] Fogal, T., Schiewe, A., and Krüger, J. (2013). An analysis of scalable gpu-based ray-guided volume rendering. *Proceedings. IEEE Symposium on Large-Scale Data Analysis and Visualization*, 2013:43–51.

[Huang et al., 2018] Huang, T.-K., Yang, C.-H., Hsieh, Y.-H., Wang, J.-C., and Hung, C.-C. (2018). Augmented reality (ar) and virtual reality (vr) applied in dentistry. *The Kaohsiung Journal of Medical Sciences*, 34(4):243–248. Special Issue on Dental Research to celebrate KMUD 60th Anniversary.

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

[LaMar et al., 2000] LaMar, E., Duchaineau, M. A., Hamann, B., and Joy, K. I. (2000). Multiresolution techniques for interactive texture-based rendering of arbitrarily oriented cutting planes. In de Leeuw, W. C. and van Liere, R., editors, *Data Visualization 2000*, pages 105–114, Vienna. Springer Vienna.

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

[Levoy, 1990] Levoy, M. (1990). Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261.

[Meyer-Spradow et al., 2009] Meyer-Spradow, J., Ropinski, T., Mensmann, J., and Hinrichs, K. (2009). Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29:6–13.

[Murphy et al., 2009] Murphy, H. A., Duchowski, A. T., and Tyrrell, R. A. (2009). Hybrid image/model-based gaze-contingent rendering. *ACM Trans. Appl. Percept.*, 5(4).

[Novotny et al., 2019] Novotny, J., Tveite, J., Turner, M. L., Gatesy, S., Drury, F., Falkingham, P., and Laidlaw, D. H. (2019). Developing virtual reality visualizations for unsteady flow analysis of dinosaur track formation using scientific sketching. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2145–2154.

[O'Leary et al., 2017] O'Leary, P., Jhaveri, S., Chaudhary, A., Sherman, W., Martin, K., Lonie, D., Whiting, E., Money, J., and McKenzie, S. (2017). Enhancements to vtk enabling scientific visualization in immersive environments. In *2017 IEEE Virtual Reality (VR)*, pages 186–194.

[Reder, 1973] Reder, S. M. (1973). On-line monitoring of eye-position signals in contingent and noncontingent paradigms. *Behavior Research Methods & Instrumentation*, 5(2):218–228.

[Reingold et al., 2003] Reingold, E. M., Loschky, L. C., McConkie, G. W., and Stampe, D. M. (2003). Gaze-contingent multiresolutional displays: An integrative review. *Human Factors*, 45(2):307–328. PMID: 14529201.

[Resnikoff, 1989] Resnikoff, H. L. (1989). *The Illusion of Reality*. Springer US, New York, NY.

[Scholl et al., 2018] Scholl, I., Suder, S., and Schiffer, S. (2018). *Direct Volume Rendering in Virtual Reality*, pages 297–302.

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

[Valois, 2000] Valois, K. D. (2000). *Seeing*. Elsevier.

[Vincent and Brannan, 2017] Vincent, P. and Brannan, R. (2017). Tobii eye tracked foveated rendering for vr and desktop.

[Vogt and Shingles, 2013] Vogt, F. P. A. and Shingles, L. J. (2013). Augmented reality in astrophysics. *Astrophysics and Space Science*, 347(1):47–60.

[Wang et al., 2018] Wang, P., Wu, P., Wang, J., Chi, H.-L., and Wang, X. (2018). A critical review of the use of virtual reality in construction engineering education and training. *International Journal of Environmental Research and Public Health*, 15(6).

[Waschk and Krüger, 2020] Waschk, A. and Krüger, J. (2020). Favr - accelerating direct volume rendering for virtual reality systems. In *2020 IEEE Visualization Conference (VIS)*, pages 106–110.

[Wolfartsberger, 2019] Wolfartsberger, J. (2019). Analyzing the potential of virtual reality for engineering design review. *Automation in Construction*, 104:27–37.

[Zimmermann et al., 2000] Zimmermann, K., Westermann, R., Ertl, T., Hansen, C., and Weiler, M. (2000). Level-of-detail volume rendering via 3d textures. In *2000 IEEE Symposium on Volume Visualization (VV 2000)*, pages 7–13.