Voronoi-Based Foveated Volume Rendering

V. Bruder 🆔, C. Schulz 🆔, R. Bauer, S. Frey 🆔, D. Weiskopf 🆔, and T. Ertl 🆔

University of Stuttgart, Germany

Abstract

Foveal vision is located in the center of the field of view with a rich impression of detail and color, whereas peripheral vision occurs on the side with more fuzzy and colorless perception. This visual acuity fall-off can be used to achieve higher frame rates by adapting rendering quality to the human visual system. Volume raycasting has unique characteristics, preventing a direct transfer of many traditional foveated rendering techniques. We present an approach that utilizes the visual acuity fall-off to accelerate volume rendering based on Linde-Buzo-Gray sampling and natural neighbor interpolation. First, we measure gaze using a stationary 1200 Hz eye-tracking system. Then, we adapt our sampling and reconstruction strategy to that gaze. Finally, we apply a temporal smoothing filter to attenuate undersampling artifacts since peripheral vision is particularly sensitive to contrast changes and movement. Our approach substantially improves rendering performance with barely perceptible changes in visual quality. We demonstrate the usefulness of our approach through performance measurements on various data sets.

CCS Concepts

• ***Human-centered computing*** → *Scientific visualization;* • ***Computing methodologies*** → *Perception;*

1. Introduction

Modern output devices are steadily increasing in pixel density and refresh rate. While this trend improves the user experience for users of large projection screens and head-mounted devices, the rendering performance of image order approaches such as volume raycasting [HKRs*06] is heavily impacted by the associated performance requirements [BMFE19]. For example, higher screen resolution typically requires a denser sampling of the image plane that can only partly be accounted for by advances in graphics hardware.

Many different approaches have been developed to improve volume rendering performance. Typically, the goal of these approaches is to reduce the rendering cost so that little to no deviations from a ground truth solution occur. In practice, empty space skipping and early ray termination are among the most widely used techniques. Our work complements these approaches by adapting the sampling strategy in different areas of an image depending on the perceptual characteristics of the human visual system. We measure user gaze using an eye-tracking system to determine the areas of the image plane that are in foveal (detailed) vision and peripheral (less sharp and colored) vision—hence, the term foveated rendering.

In this work, we focus on the specifics of foveated volume raycasting. Our contribution can be broken down into the modeling, realization, and evaluation of a foveated volume rendering system: We pre-compute a sampling mask based on visual acuity fall-off using the Linde-Buzo-Gray algorithm, shift this mask according to user gaze, reconstruct the image based on Voronoi cells using natural neighbor interpolation, and apply temporal smoothing to make undersampling artifacts less disturbing.

2. Related Work

Volume Rendering Volume rendering has many application domains such as medical imaging, visualization of scientific simulation data, and visual arts as in computer-generated imagery. GPU raycasting has established itself as the de-facto standard for performing real-time direct volume rendering in workstation environments [RGW*03, SSKE05, HKRs*06]. Despite significant advances in computing capabilities of GPUs, volume rendering stays a computationally expensive task due to growing data set sizes and increasing pixel densities of output devices. Therefore, improving performance in various scenarios through different approaches has emerged into an active field of research [BHP15] such as data oriented techniques [HAAB*18], progressive methods [FSME14], and prediction-based techniques [BFE17].

Foveated Rendering The limitations and modeling of human perception for perception-driven rendering is an active field of research. However, most methods focus on object order rendering or ray tracing techniques [WSR*17]. For instance, Guenter et al. [GFD*12] present foveated rendering for rasterization using GPUs. They use three images with different resolutions and level of details, leading to an average performance improvement of factor six. Stengel et al. [SGEM16] adapt the shading in a deferred rasterization pipeline to incorporate perceptual aspects into real-time rendering. Besides simulating the acuity fall-off using gaze, they also adapt their image-space sampling scheme to make use of other physiological factors such as contrast sensitivity and eye movement. With this approach, they could reduce the fragment shading costs by 50% to 80%. Other works focus on perception-driven acceleration for ray-
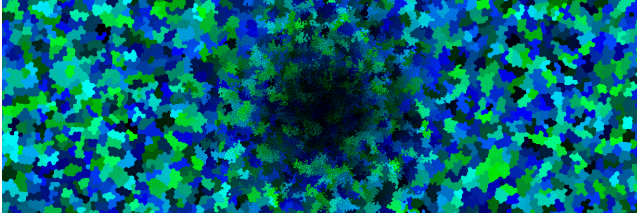
(a) *Color-coded Voronoi cells and corresponding ray origins in black.*
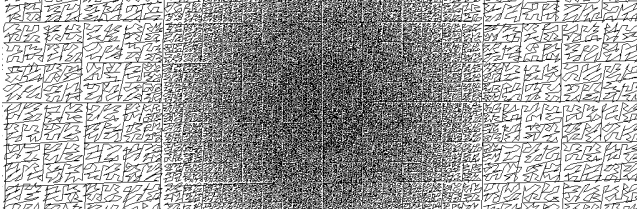
(b) *Ray origins connected by lines to illustrate the Morton ordering.*

Figure 1: *Illustration of the sampling mask for volume raycasting.*

tracing techniques [Mit87, JIC*09]. In their work from 1990, Levoy and Whitaker [LW90] present a gaze-directed volume renderer for the Pixel-Planes 5 rendering engine. They reduce the sampling rate in the image as well as object space, thereby generating a performance improvement of around factor five. In comparison, our approach leverages modern graphics hardware and features optimized sampling strategies in image and object space.

3. Visual Acuity Fall-Off Model

Many works investigated and confirmed the fall-off of the visual acuity in the periphery [SRJ11]. Inverting the acuity results in the *minimum angle of resolution* (MAR) that can be approximated by a linear model [Wey63]:

$$MAR = \omega_0 + m \cdot e, \tag{1}$$

where ω_0 denotes the smallest resolvable angle, e the eccentricity in degrees, and m the slope. The MAR model has been confirmed to match performance results in low-level vision tasks as well as anatomical features of the eye. Adaptions thereof have been successfully employed in different foveated rendering methods [GFD*12, SGEM16, VST*14]. We approximate this hyperbolic acuity fall-off function using a 2D Gaussian function depending on screen resolution, size, approximated viewing distance, and estimated photoreceptor distribution. Our photoreceptor topology estimation is based on an average foveal acuity for healthy adults below the age of 50 [EYW95]. In addition, a smooth function results in a more pleasant sampling mask (cf. Figure 1a). Based on our acuity fall-off function around the gaze, we now can infer the number and spread of casted rays in screen space as well as the sampling density along the rays in object space.

4. Foveated Volume Rendering

Our technique carries out several (computationally intensive) pre-processing steps, minimizing the induced cost at runtime. During the first pre-processing step, we compute a sampling mask to determine the starting points of the rays for raycasting using the Linde-Buzo-Gray (LBG) stippling algorithm (cf. Figure 1a, Sec. 4.1). The

algorithm has several advantages over a simple sampling strategy: it arranges the rays' starting positions in such a way that little or no visible patterns manifest that could irritate the viewer [DSZ17]. According to our observations, this property translates well to foveated volume rendering. Also during pre-processing, we optimize the spatial locality of sampling rays by sorting them according to Morton order, also called Z-curve (cf. Figure 1b). This improves ray locality and therefore caching behavior when using the generated sampling masks for volume raycasting at runtime (Sec. 4.2). The sampling mask has at least twice the size of the screen resolution since we translate it during volume rendering according to the gaze, effectively keeping the high-density part in the middle of the texture at the foveal region of the user.

Our sampling method (in image space) is based on Voronoi cells, just like natural neighbor interpolation, which provides a mathematical symmetry between sampling and reconstruction strategy. Accordingly, we have modified the LBG algorithm to compute neighbors and weights for natural neighbor interpolation of the sparsely sampled rays during pre-processing (Sec. 4.3). These can then be used directly during runtime for reconstruction. Finally, we apply a temporal smoothing filter to attenuate undersampling artifacts in peripheral vision (Sec. 4.4).

4.1. Weighted Linde-Buzo-Gray Algorithm

The objective of the Linde-Buzo-Gray algorithm [DSZ17, GSS*19] is to arrange stipples depending on a function. In this work, we want to arrange starting positions of representative rays, instead of stipples, according to a density function over the sampling mask (cf. Sec. 3). Let $\Phi : \mathbb{R}^2 \to [0,1]$ be a function that maps pixel coordinates to sampling density. The algorithm is initialized with a random distribution of ray positions. During each iteration, the neighborhood of each ray position $r \in R$ is inferred from the Voronoi diagram to assess how well each ray represents its proximity in Φ by integrating over the corresponding Voronoi cell V_r. Formally, the target density for a ray position representing the cell T_r is defined as:

$$T_r = \iint\limits_{V_r} \Phi(x,y) \, dA \, . \tag{2}$$

In our case, this is the same as accumulating the density over all pixels of the sampling mask that are part of to the respective Voronoi cell. Then, the algorithm compares the cost function T_r with the area occupied by the ray position A_r to measure the error ε. The ray positions are then adjusted according to one of three cases: (1) move and relax ray position if $(T_r \in [A_r - \varepsilon, A_r + \varepsilon])$, (2) split ray if $(T_r > A_r + \varepsilon)$, or (3) delete ray if $(T_r < A_r - \varepsilon)$. This process is repeated until the error is below a given threshold, i.e., the amount of rays remains unchanged. To stabilize iteration, we use a hysteresis function $A_r \cdot (1 \pm (\varepsilon_0 + i \cdot \varepsilon_\Delta) / 2)$ for the lower and upper bound of the cases above with $\varepsilon_0 = 0.5$ and $\varepsilon_\Delta = 0.1$ in the ith iteration.

4.2. Volume Raycasting

Our GPU-based volume raycaster samples 3D textures using perspective projection, performing front-to-back compositing using a post-classification model with linear transfer functions. Density values that are sampled along casted rays are determined using tri-linear interpolation. The renderer features local Phong illumination, based on gradients (central differences), early ray termination, and

empty space skipping for acceleration. We adjust the distance between samples as given by our sampling mask proportional to our acuity fall-off function, not only in image but also in object space. Akin to a decreased sampling density in image space that is a direct result of the visual model (Sec. 3), a coarser sampling along rays produces a lower resolution approximation with respect to a reference image (e.g., [BMWM06]).

4.3. Natural-Neighbor-Based Image Reconstruction

To reconstruct an image of the volume, we have to perform an interpolation of the sparsely sampled screen space. We chose a natural neighbor interpolation scheme [Sib80] because it provides a smooth approximation, requires only local neighbors, and is generally C_1 continuous. Moreover, it fits our sampling strategy well since it is also Voronoi-based (cf. Sec. 4.1).

Despite being computationally expensive, the method has been used in different application domains such as engineering, mechanics, and also in scientific visualization [PLK*06]. To compute the interpolated value at a given point, a new Voronoi cell is inserted into the existing tessellation at the position (x, y) of the point. The estimate G of the new point is then calculated by using the areas A of the intersections with the neighboring cells in relation to the total area of the new cell as weighting factors for the interpolation:

$$G(x,y) = \sum_{i=0}^{k} \frac{A(S_i \cap N)}{A(N)} \cdot f(x_i, y_i), \qquad (3)$$

with $A(N)$ being the area of the new cell N and $f(x_i, y_i)$ being the known values at the k neighboring cells S_i.

Computing the weights is offloaded into a pre-processing step that results in two textures: Both textures have twice the resolution of the screen to accommodate for gaze-dependent translation of the mask. One texture stores the indices of the neighboring cells. The other texture stores the weights that are used for interpolation. This design makes it simple and efficient to compute the interpolations on a per-pixel basis after the raycasting.

4.4. Temporal Smoothing Filter

While natural neighbor interpolation provides a precise and smooth reconstruction, sparse sampling introduces aliasing artifacts, especially at hard transitions in the volume data. In addition, undersampling artifacts may occur especially near fine structures due to the low sampling density in peripheral vision. However, peripheral vision is particularly sensitive to contrast changes and movement. Therefore, we attenuate those artifacts using a temporal smoothing filter by averaging between n previous frames. Here, we found $n = 8$ to be a good compromise between sufficient smoothing, perceived fade, and perceived frame rate [DoV17]. Effectively, the filter also provides a form of anti-aliasing through implicit supersampling. Moreover, it hides the transition from blurry to sharp during rapid eye movements that can be noticeable if the screen update lags behind the eye movement. This approach also helps with eye-tracking devices with low sampling rate. To avoid introducing a motion blur effect, we do not apply the temporal smoothing during image-altering changes such as camera manipulation and transfer function modifications.

Table 1: *Rendering performance.*

Data set	Mean frame rate		Relative speedup
	Regular	Foveated	0 1 2 3 4 5
Combustion	73.85	154.20	⊢─⊏⊐─┤
Supernova	37.60	105.28	⊢─⊏⊐─┤
Vortex cascade	33.27	104.87	•⊢⊏⊐─┤
Zeiss	98.50	177.09	⊢─⊏⊐─┤
Flower	22.79	74.08	⊢──⊏⊐──┤
Chameleon	35.99	99.78	∘ ∞∞⊢⊏⊐─┤ ∞

5. Results

We tested our approach using a stationary Tobii Pro Spectrum eye-tracker with a sampling rate of 1200 Hz. For runtime performance evaluation, we simulated deterministic and randomly scattered gaze positions for comparability and reproducibility. All images were rendered with a resolution of 1024×1024 px on a 1080 p screen with a 24" diagonal, using a workstation equipped with an Intel Core i7-7700K, 32 GB RAM, an NVIDIA GeForce GTX 1070 with 8 GB of VRAM, and running Windows 10 (October 18 Update). Our single-pass implementation of volume raycasting uses OpenCL 1.2 for reasons of device and platform portability. The evaluated pixel colors are written to a texture and shown with a screen-filling quad using OpenGL. The sampling rate along the rays was set to be twice the data resolution (i.e., a sampling distance of half a voxel) as base value. We used CT-scans (Chameleon, Zeiss car component, flower) and individual time steps from simulation data (vortex cascade, combustion, entropy of a supernova). For each data set, we designed one specific transfer function that we used across our measurements shown in Figure 2. To give an impression of the sampling pattern, Figure 2h shows the same configuration as Figure 2g but without natural neighbor interpolation applied, i.e., only pixels containing ray starting positions are colored. For comparison, Figure 2f shows the reference without foveated rendering applied.

Following recommendations by Bruder et al. [BMFE19], we measure 256 random camera configurations, rotating around the volume and changing the distance to the volume, while the target of the camera is the center of the data set (the maximum distance is set to the minimum distance required to fit the whole projected volume onto the screen for any camera configuration). For each camera configuration, we test 256 random gaze positions, resulting in a total of 65536 measurement configurations. We measure five frames for each configuration and keep the median frame time as representative value. Then, we calculate the mean frame rate and relative speedup based on those representative measurements. Table 1 shows frame times with and without foveated rendering as well as the relative speedup for each data set. On average, we achieved speedups between factor 1.8 and 3.2 using our foveated rendering, depending on the data set and transfer function. Generally, the speedup is higher for volumes with less empty space. We could barely experience perceptible changes with respect to visual quality. However, quantifying perceived image quality in a lab or crowd sourcing study as suggested by Hosu et al. [HHZS16] is beyond scope of this work.

The reduced number of rays when using our approach (less than 8%) may indicate an even bigger performance gain. However, the
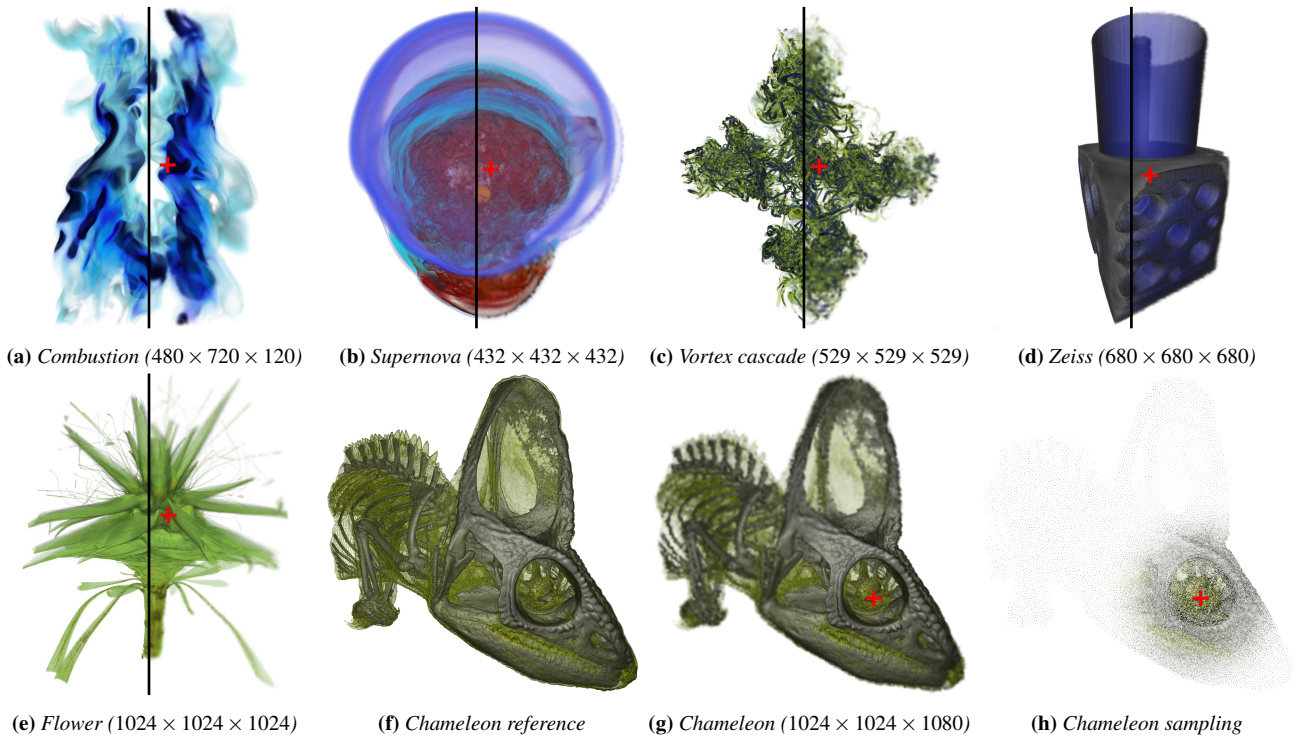
(a) *Combustion ($480 \times 720 \times 120$)* **(b)** *Supernova ($432 \times 432 \times 432$)* **(c)** *Vortex cascade ($529 \times 529 \times 529$)* **(d)** *Zeiss ($680 \times 680 \times 680$)*

(e) *Flower ($1024 \times 1024 \times 1024$)* **(f)** *Chameleon reference* **(g)** *Chameleon ($1024 \times 1024 \times 1080$)* **(h)** *Chameleon sampling*

Figure 2: *Example renderings of each data set with transfer functions from the benchmark. The left sides show the reference renderings, the right ones our foveated renderings with a red cross indicating the respective fixation point. For the Chameleon (g), we also provide sampling results before reconstruction (h).*

sparse regions in the periphery often correspond to those rays accelerated by empty space skipping. Furthermore, we cannot take advantage of caching as efficiently compared to using the regular volume rendering due to the non-consecutive voxel-access pattern (cf. Figure 2h). Additionally, there is the small overhead of natural neighbor interpolation, which amounts to 1.5 ± 0.157 ms for all data sets, as well as the general overhead (kernel invocation, etc.).

6. Discussion and Future Work

We approximate the acuity fall-off in a rather coarse fashion by using a 2D Gaussian function. While our approximation of the acuity fall-off by using a 2D Gaussian function is sufficiently accurate for people with normal vision, individual differences exist regarding the distribution of rod and cone cells in the retina. Thus, calibrating our system according to the physiology of individuals could improve the quality and performance of foveated rendering, as could further optimization based on user feedback.

The coarse sampling in the peripheral regions can further lead to undersampling artifacts, especially near sharp borders or fine structures in the data. While our temporal smoothing filter helps avoid such artifacts, it may potentially induce undesired motion blur effects. Therefore, we have refrained from using the filter for interaction and dynamic data. Here, we found that explicit and expected movement tends to make rendering artifacts less prominent. We have also explored a 3D mipmap stack and linear interpolation between the levels based on the sampling density. However, the performance hit caused by additional 3D texture fetches was too

high compared to the improvement in image quality. In general, we suppose that our implementation could be further optimized to yield even better speedups from the significant reduction in emitted rays.

High frame rates are crucial in virtual reality to prevent dizziness, e.g., when using head-mounted displays. We could achieve substantial performance improvements using foveated rendering. Therefore, we plan to port our technique to head-mounted displays with integrated eye-tracking devices in the future. In this context, we are further interested in the performance characteristics of stereo rendering with respect to caching.

7. Conclusion

We presented an approach that utilizes the acuity fall-off in the visual system to accelerate volume rendering. To this end, we modified a typical volume raycaster to adapt various sampling parameters to the gaze of the user. For the approximation of sample density and reconstruction during rendering, we propose a novel technique based on the Linde-Buzo-Gray algorithm and natural neighbor interpolation. Using a conservative acuity fall-off function, we could achieve average speedups between 1.8 and 3.2 for different data sets, with hardly perceptible changes in image quality in peripheral areas.

References

[BFE17] BRUDER V., FREY S., ERTL T.: Prediction-based load balancing and resolution tuning for interactive volume raycasting. *Visual Informatics 1*, 2 (2017), 106–117. 1

[BHP15] BEYER J., HADWIGER M., PFISTER H.: State-of-the-art in GPU-based large-scale volume visualization. *Computer Graphics Forum 34*, 8 (2015), 13–37. 1

[BMFE19] BRUDER V., MÜLLER C., FREY S., ERTL T.: On evaluating runtime performance of interactive visualizations. *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1. 1, 3

[BMWM06] BERGNER S., MÖLLER T., WEISKOPF D., MURAKI D. J.: A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1353–1360. 3

[DoV17] DoVALE E.: High frame rate psychophysics: Experimentation to determine a jnd for frame rate. *SMPTE Motion Imaging Journal 126*, 9 (2017), 41–47. 3

[DSZ17] DEUSSEN O., SPICKER M., ZHENG Q.: Weighted Linde-Buzo-Gray stippling. *ACM Transactions on Graphics 36*, 6 (2017), 1–12. 2

[EYW95] ELLIOTT D. B., YANG K., WHITAKER D.: Visual acuity changes throughout adulthood in normal, healthy eyes: seeing beyond 6/6. *Optometry and Vision Science: Official Publication of the American Academy of Optometry 72*, 3 (1995), 186–191. 2

[FSME14] FREY S., SADLO F., MA K.-L., ERTL T.: Interactive progressive visualization with space-time error control. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2397–2406. 1

[GFD*12] GUENTER B., FINCH M., DRUCKER S., TAN D., SNYDER J.: Foveated 3D graphics. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 164. 1, 2

[GSS*19] GÖRTLER J., SPICKER M., SCHULZ C., WEISKOPF D., DEUSSEN O.: Stippling of 2D scalar fields. *IEEE Transactions on Visualization and Computer Graphics* (2019). 2

[HAAB*18] HADWIGER M., AL-AWAMI A. K., BEYER J., AGUS M., PFISTER H.: SparseLeap: Efficient empty space skipping for large-scale volume rendering. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 974–983. 1

[HHZS16] HOSU V., HAHN F., ZINGMAN I., SAUPE D.: Reported attention as a promising alternative to gaze in iqa tasks. In *Proc. 5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)* (2016), pp. 117–121. 3

[HKRs*06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D., ENGEL K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006. 1

[JIC*09] JIN B., IHM I., CHANG B., PARK C., LEE W., JUNG S.: Selective and adaptive supersampling for real-time ray tracing. In *Proceedings of the Conference on High Performance Graphics* (2009), HPG '09, ACM, pp. 117–125. 2

[LW90] LEVOY M., WHITAKER R.: Gaze-directed volume rendering. *SIGGRAPH Computer Graphics 24*, 2 (1990), 217–223. 2

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. *SIGGRAPH Computer Graphics 21*, 4 (1987), 65–72. 2

[PLK*06] PARK S. W., LINSEN L., KREYLOS O., OWENS J. D., HAMANN B.: Discrete Sibson interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 2 (2006), 243–253. 3

[RGW*03] ROETTGER S., GUTHE S., WEISKOPF D., ERTL T., STRASSER W.: Smart hardware-accelerated volume rendering. In *Proceedings of the Symposium on Data Visualisation (VISSYM)* (2003), pp. 231–238. 1

[SGEM16] STENGEL M., GROGORICK S., EISEMANN M., MAGNOR M.: Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum* (2016), vol. 35, pp. 129–139. 1, 2

[Sib80] SIBSON R.: A vector identity for the Dirichlet tessellation. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1980), vol. 87, Cambridge University Press, pp. 151–155. 3

[SRJ11] STRASBURGER H., RENTSCHLER I., JÜTTNER M.: Peripheral vision and pattern recognition: A review. *Journal of vision 11*, 5 (2011), 13–13. 2

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *In Proceedings of the Fourth International Workshop on Volume Graphics* (2005), pp. 187–241. 1

[VST*14] VAIDYANATHAN K., SALVI M., TOTH R., FOLEY T., AKENINE-MÖLLER T., NILSSON J., MUNKBERG J., HASSELGREN J., SUGIHARA M., CLARBERG P., ET AL.: Coarse pixel shading. In *Proceedings of High Performance Graphics* (2014), Eurographics Association, pp. 9–18. 2

[Wey63] WEYMOUTH F. W.: Visual sensory units and the minimum angle of resolution. *Optometry and Vision Science 40*, 9 (1963), 550–568. 2

[WSR*17] WEIER M., STENGEL M., ROTH T., DIDYK P., EISEMANN E., EISEMANN M., GROGORICK S., HINKENJANN A., KRUIJFF E., MAGNOR M., ET AL.: Perception-driven accelerated rendering. *Computer Graphics Forum 36*, 2 (2017), 611–643. 1