EG UK Computer Graphics & Visual Computing (2024)
A. Slingsby and D. Hunter (Editors)

Skipping Spheres: SDF Scaling & Early Ray Termination for Fast Sphere Tracing

A. Polychronakis[1], G. A. Koulieris[2] and K. Mania[1]

[1]TU of Crete, School of Electrical & Computer Engineering, Greece
[2]Durham University, Department of Computer Science, United Kingdom

Abstract

This paper presents a rapid rendering pipeline for sphere tracing Signed Distance Functions (SDFs), showcasing a notable boost in performance compared to the current state-of-the-art. Existing methods endeavor to reduce the ray step count by adjusting step size using heuristics or by rendering multiple intermediate lower-resolution buffers to pre-calculate non-salient pixels at reduced quality. However, the accelerated performance with low-resolution buffers often introduces artifacts compared to fully sphere-traced scenes, especially for smaller features, which might go unnoticed altogether. Our approach significantly reduces steps compared to prior work while minimising artifacts. We accomplish this based on two key observations and by employing a single low-resolution buffer: Firstly, we perform SDF scaling in the low-resolution buffer, effectively enlarging the footprint of the implicit surfaces when rendered in low resolution, ensuring visibility of all SDFs. Secondly, leveraging the low-resolution buffer rendering, we detect when a ray converges to high-cost surface edges and can terminate sphere tracing earlier than usual, further reducing step count. Our method achieves a substantial performance improvement (exceeding 3$\times$ in certain scenes) compared to previous approaches, while minimizing artifacts, as demonstrated in our visual fidelity evaluation.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Rendering;***

1. Introduction

Given the recent advancements in computer graphics hardware and software, particularly the emergence of ultra-high resolution displays [KAS*19] and Virtual Reality, there is an increasingly pressing demand for fast and efficient rendering algorithms. In this pursuit, Signed Distance Functions (SDFs) have emerged as invaluable tools [Don05], offering mathematical representations that describe geometric shapes with high precision. By determining the distance from any point in space to the nearest point on the -mathematically described- shape's surface, SDFs facilitate rendering tasks, from ray tracing to collision detection [FLM24, MEM*20], while also enabling procedural content generation due to their efficient memory utilization [VSJ22], as they do not require storing geometry data.

Rendering objects using SDFs presents a choice between polygonisation [GVPN09, FPSM06], which reduces accuracy and demands substantial computational resources [FLM24, MEM*20], and the more precise and efficient method of Sphere Tracing [Har96]. However, when rendering scenes by sphere tracing SDFs, computational complexity rises significantly when dealing with complex objects and surfaces. This is especially noticeable near sharp boundaries of scene elements, where the traced rays have to repeatedly make smaller steps to accurately approach the nearest SDF, an inherent characteristic of how sphere tracing SDFs works. As a result, a large number of computational steps are required to ensure convergence onto the object surface (Figure 3a). But unlike ray-tracing [Whi79], which benefits from dedicated hardware-level acceleration structures in GPUs [PFL*13], similar hardware acceleration for sphere tracing SDFs is currently lacking, as it has only been tested experimentally [SN09, Lik08, FJL*17].

Thus, rendering complex SDF-based scenes in real time, requires efficient sphere tracing algorithms. Several strategies have been used to accelerate sphere tracing. Some methods reduce the ray step count by adjusting the step size using heuristics, e.g., dynamic step size adjustment [KSK*14] or by rendering multiple intermediate lower-resolution buffers to pre-calculate non-salient pixels at reduced quality [PKM23]. Additionally, methods such as cone tracing and super sample anti-aliasing have improved rendering quality and efficiency [BBV19, CZR*23]. Local Lipschitz bounds [GGPP20] and grid density fields [SEAM22] have also shown promising results in reducing computational overhead. However, these methods often introduce rendering artifacts, usually as a trade-off between rendering quality and speed.

In this paper, we further accelerate sphere tracing while min-

DOI: 10.2312/cgvc.20241219

imising artifacts in a two-step process that employs a single low-resolution buffer (to reduce initial steps), based on two key observations: Firstly, by performing SDF scaling, i.e., making the surface/object larger, a simple mathematical operation, within the low-resolution buffer, we effectively expand the reach of implicit surfaces during rendering in low resolution, guaranteeing the visibility of even the smallest SDFs/objects that might have be missed otherwise. Secondly, we can identify instances where a ray converges towards high-cost surface edges, allowing us to terminate sphere tracing prematurely in the low resolution buffer, thereby significantly decreasing the step count. Full-resolution sphere tracing is then performed using the estimated starting points, thus placing rays closer to the traced surfaces, resulting in a massive reduction of sphere tracing steps, whilst minimising artefacts due to missed surfaces. Our method supports all types of lights and the acceleration technique can be applied both to primary rays and shadow rays, enabling accelerated soft shadows.

Our quality evaluation demonstrated that our method renders high-resolution images nearly identical to the ground truth full-step sphere tracing, with a speedup of up to 3× in sphere tracing performance, while minimizing artifacts, as demonstrated in our visual fidelity evaluation (up to 60dB). Our contributions include:

- We introduce a rapid rendering pipeline for sphere tracing SDFs, based on SDF scaling and early ray termination when approaching surfaces.
- We extend our method to support accelerated shadow rays and soft shadows.
- We perform a thorough evaluation of our method, including an ablation study for its individual components, comparing it to the state-of-the-art. Our evaluation indicated a substantial performance improvement over previous work, while minimising artifacts.

2. Related Work

In this section, we provide an overview of SDFs, scene representation using SDFs, sphere tracing SDFs and shadows, and introduce past research on sphere tracing optimisations.

2.1. Singed Distance Functions

Signed Distance Functions are mathematical representations primarily used in computer graphics to describe geometric shapes. They determine the distance from a point in space to the closest point on the surface of the shape, with the sign indicating whether the point is inside or outside the shape. Mathematically, for a point $\mathbf{p}$ in space and a shape with surface defined by $S(\mathbf{x}) = 0$, where $\mathbf{x}$ is a point on the surface, the signed distance function is given by $d(\mathbf{p}) = \text{sign}(S(\mathbf{p}))\|S(\mathbf{p})\|$, where $\|\cdot\|$ denotes the Euclidean norm. Functions of this type enable shape manipulation and rendering, resulting to efficient algorithms for tasks such as ray tracing, sphere tracing, and collision detection in computer graphics [FLM24, MEM*20]. SDFs are also used to manually or procedurally generate content [HSK89] due to their efficient memory utilization. Examples include procedural displacement mapping tools [Don05], artistic rendering [Ini22], or fully connected neural networks learning from multiple images [MST*20, OPG21, PCP-MMN21] to estimate surfaces in those images.

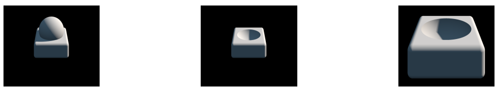

(a) *Union* **(b)** *Subtraction* **(c)** *Scaling*

Figure 1: *Useful operations for handling / combining SDFs. (a) Union of 2 SDFs (box + sphere), b) Subtraction of 2 SDFs (box - sphere) and c) Scaling of (b).*

2.2. Scene Representation

To create a 3D scene, each object is represented by a single or more SDFs or by combining SDFs using operators. For example, this equation describes a sphere: $d_{sphere}(p,o,r) = \|p-o\| - r$, where p is the current ray position in the world, o is the center of the sphere, and r is the size of the sphere. By combining primitive SDFs, such as the one above, using the union or subtraction operators (Figure 1), we can create more complex shapes. The union operation takes the calculated distance of two SDFs and chooses the minimum distance.

Using the subtraction operation, we can remove parts of the implicit surface (Figure 1b). Subtraction works similarly to union, but instead inverts the sign of the distance, i.e., object we want to remove from the other SDF as seen in the this equation: $d(p,o_1,o_2,r) = max(d_{sphere}(p,o_1,r), -d_{sphere}(p,o_2,r))$, and contrary to union, takes the maximum distance.

Another common operation when creating a 3D environment is surface scaling to adjust the size of objects. To scale an SDF, we can compress the space where the surface exists. Distance estimation is then performed in the *scaled space* and finally we multiply the result with the scale value to rescale the space as seen in this equation: $d_{sphere}(p) = (\|(p-o)/s\| - r) * s$, where s is the scaling factor applied in the signed distance functions (Figure 1c).

2.3. Rendering SDFs

Rendering objects using SDFs can be achieved in two ways. One method involves polygonizing the implicit surface of the object [NY06] i.e., converting the mathematical representation of the object into a polygonal mesh. Although this enables the use of conventional rendering techniques, such as rasterization or ray tracing, it often compromises accuracy and requires significant memory and computational resources.

Alternatively, Sphere Tracing can be used for more accurate real-time rendering. Hart introduced sphere tracing [Har96] for ray tracing implicit surfaces like fractals [HSK89, Kim15, MEG04] described by SDFs. In sphere tracing, a ray propagates in 3D space by calculating the distance between said ray and each surface of the 3D environment (Figure 3a). When the minimum distance is found, the ray will move forward using that distance as a step in the original direction the ray was shot.

A sphere tracing process is terminated if the ray a) approaches a surface closer than a threshold *epsilon* - the smaller the threshold, the more accurately the surface is approximated, but on the flip side, the higher the required step count, b) surpasses the maximum scene travel distance t_{max}, or c) performs a maximum number of steps s_{max}. Mathematically, a ray in sphere tracing is expressed as: $p_t = r_o + r_d * t$, where r_o is the ray origin, r_d is the ray direction and t is the ray's endpoint. The endpoint is increased iteratively during tracing and can be expressed as: $t = t + d(p_t)$, where $d(p_t)$ is the function that returns the minimum distance from the SDF.

We can exploit the immediate access to global scene information provided by SDFs, enabling the rendering of realistic soft shadows. Simply by considering the distance to the nearest surface and the distance along the ray during ray marching, a penumbra factor can be computed for each step point, resulting in shadows that are sharper near occluder contact and softer away from it [Aal18, Ini22].

2.4. Accelerating Sphere Tracing

As scene complexity increases (*more* SDFs describing the scene), the sphere tracing step size generally *decreases*, as there are more surfaces *closer* to the ray's endpoint increasing the step size, but also, generally, more SDF distance evaluations are also required. As a result, a ray will need to be traced for significantly more steps until it converges to a surface, increasing computations overall and resulting in a drop in performance. To achieve real-time rendering, several strategies aimed at accelerating sphere tracing have been presented:

The first family of strategies aims at reducing the required number of steps. Keinert et al. [KSK*14] controlled step advancement by multiplying the step size using a scene-specific fixed multiplier ranging between 1 and 2, which enhances tracing speed but the multiplier has to be manually selected. This scene-specific configuration limitation prompted subsequent research by Bán et al. [BV23], to propose a dynamic step adjustment based on a function including the rate of change of the previous step size and the current step. Bálint et al. [BV18] further refined [BV23] by constructing linear distance approximations of rays to SDFs; the method is particularly effective for planar surfaces. Cone tracing by Bán et al. [BBV19] and super sample anti-aliasing by Chubarau et al. [CZR*23] have further improved rendering quality and efficiency. A method utilising local Lipschitz bounds [GGPP20], has demonstrated promising results in reducing computational overhead by avoiding calculating distances that don't intersect with a ray. All aforementioned approaches work well for older GPUs, that had large batch sizes, meaning that pixels that need to perform more steps would force pixels that needed less steps to stall. For modern GPUs, these methods provide limited performance improvements while still generating artifacts (see Evaluation section 4).

The second family of strategies leverages acceleration structures to accelerate ray traversal towards scene points that contain surfaces. Söderlund et al. [SEAM22] and Balint et al. [BK21] propose the use of a grid density field where each voxel of the grid has a pre-calculated signed distance to the nearest SDF at its eight corners. They then compute the distance between a ray and the surface defined by trilinear interpolation of signed distances at corners of a voxel. The major drawback of these approaches is that the rendering quality and increased memory usage of the produced volumetric grid relies on a step size for the grid that depends on the scene's structure and complexity.

The third family employs low resolution render targets [PKM21] to approximate ray distances in the 3D environment. Polychronakis et al. [PKM23], combine several buffers in a pyramidal render target hierarchy and filtering to spawn rays closer to surfaces resulting in fewer steps overall. This method offers a balance between computational resources (less memory than grids) and similar rendering quality while being faster than the other methods, however, it suffers from artifacts and requires a complex hierarchy of render targets (high memory cost) and filtering (high computational cost).

In the next section, we present a method for both improving sphere tracing performance, and reducing rendering artifacts.

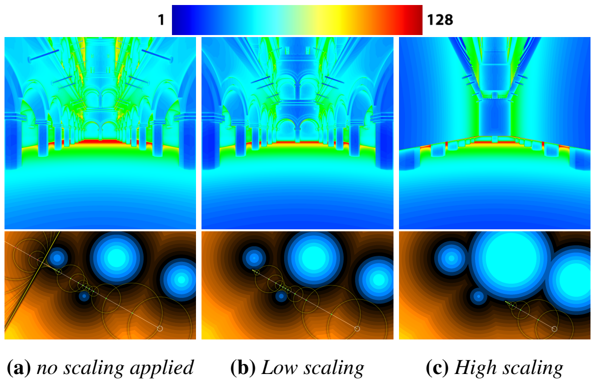
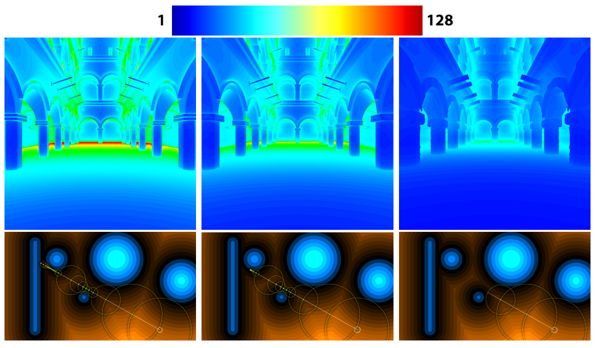


Figure 2: *SDF scaling steps (1-128, visualised) for different parameters in the Sponza atrium (top) and a schematic of the phenomenon (bottom). Left: No scaling applied in the low resolution buffer and small surfaces are missed. Middle: SDF size was increased leading to small surfaces been correctly "hit". Right: Extreme SDF scaling leads to artifacts.*

3. SDF Scaling and Early Ray Termination for Fast Sphere Tracing

3.1. Method Overview

Our method draws inspiration from prior techniques aimed at reducing ray steps by adjusting step size and generating lower resolution renders [PKM23]. Yet, the speed gained from low-resolution buffers can lead to noticeable artifacts, particularly in small features, necessitating filtering. Our approach minimizes artifacts while significantly cutting down steps compared to previous methods. This is achieved by using a single low-resolution buffer and a novel tracing optimisation instead of a filtered pyramid. This provides a *head start* origin for bundles of rays when rendering at the full resolution, by positioning rays closer to the surfaces using the information stored in this low resolution buffer, significantly improving performance. First, we perform SDF scaling in the low-resolution buffer, effectively enlarging the footprint of the implicit surfaces when rendered in low resolution, ensuring visibility of all SDFs; Secondly, we identify when a ray converges to high-cost surface edges and can terminate sphere tracing earlier than usual, further reducing step count significantly. Our method works in two passes described below.



(a) *Low ϵ* **(b)** *Medium ϵ* **(c)** *High ϵ*

Figure 3: *Step visualisation for different ϵ values in the Sponza atrium (top) and a schematic of the phenomenon (bottom). Step count is reduced from left to right as the ϵ value is increasing.*

3.2. Pre-processing: Low Resolution

In the pre-prossessing pass, we render primary and shadow rays in a low resolution buffer (for a 4K output buffer we reduce the low resolution buffer to a quarter of the resolution in each axis, i.e., from 3840x2160px to 960x540px) while applying scaling operations in the implicit surfaces and use a larger *epsilon*, i.e., terminate the ray further back from the surface, storing a smaller distance in the low-resolution buffer (Figure 3). We scale SDFs uniformly because we have no prior knowledge of the volume surface and we increase the volume of the SDFs using the scale operator.

Before scaling the SDFs, we establish whether the camera distance is positive or negative (meaning it is in front or behind an implicit surface, or inside or outside an implicit volume if it is a closed shape). If the distance is positive we scale up, i.e., inflate the volume surface of the SDF. If the distance is negative, we scale down the volume surface, i.e., deflate the SDF. This estimation has negligible computational cost.

This effectively expands the reach of implicit surfaces during rendering in low resolution, guaranteeing the visibility of even the smallest SDFs/objects that might have be missed otherwise. We store the traversed distance in the low-resolution buffer providing a head start origin for bundles of rays at the full resolution (for our target resolution each pixel will be reused by groups of 4x4 pixels (16) in the full resolution) by positioning rays closer to the surfaces using the information stored in the low-resolution buffer. This significantly enhances performance by reducing the distance rays need to travel before reaching surfaces.

For objects that have holes (generated using the subtraction operation) instead of increasing the volume size of the SDF that generates the hole, we *decrease* it , to improve the approximation. This works only when the camera is in front of the implicit surfaces: if the camera is behind the implicit surfaces, we scale up the SDF.

This approach once again, forces the SDFs to cover *more* pixels in the lower resolution to get a better estimation of the distance to be used in the second stage (Figure 2). When a larger surface overlaps with a smaller one, this provides a relaxed estimation of the distance between the ray and the surface (Figure 2c).

Early Ray Termination. The scaled SDFs may overlap with one another as they now have a different footprint than they originally had. To avoid surface overlaps with a smaller SDF, we terminate the sphere tracing process when the distance between a ray and the surface is smaller than a modified, *larger* epsilon value than normally used. This ensures that the ray doesn't bypass the edges of surfaces and provides a fast distance approximation. The drawback of this approach is that when we set a higher epsilon, shapes with edges start to become more rounded eliminating the edges and providing an inaccurate approximation (Figure 3c).

3.3. Main pass: Full-resolution

In the second pass, we perform sphere tracing at full resolution (4K in our system), for both primary and shadow rays [Har96] to produce the final frame. We sample the approximated buffer to get the approximated distance, and we set the origin of each ray in the 3D world by reconstructing its position based on the distance stored in the low resolution buffer - instead of the camera origin. We then perform sphere tracing for each pixel. This results in rays performing significantly fewer steps compared to the ground truth sphere tracing (Figure 5). In our method, we take advantage of the additional computational resources by in turn spawning more rays to enable anti-aliasing without increasing the overall frame time cost. For the second, high resolution pass of soft shadow tracing, we use the approximated distance to trace the shadows and set a *shadow hardness* k value according to the effect we want to achieve (higher k value "harder" shadows) (Figure 6).

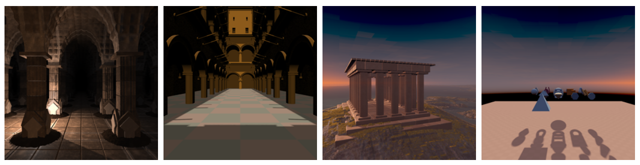


Figure 4: *The scenes used in the evaluation, by Inigo Quilez [Ini22] and [CZR*23].*

4. Evaluation

4.1. Overview

We conducted a performance and visual fidelity assessment of our system, our primary objective being twofold: to demonstrate the superior efficiency of our method and to validate that our rendering outputs attain comparable (or better) visual quality to the state-of-the-art and ground truth sphere tracing. We compare our method for primary rays against the Enhanced Sphere tracing [BV18], Auto-Relaxed Sphere Tracing [BV23], Inverse Pyramid Rendering (IPR) [PKM23] and classic sphere tracing [Har96] which we consider the ground truth. We compare our method for shadow rays (soft shadows) against all aforementioned method excluding IPR as it does not support shadows. To ascertain the significance of key components within our rendering pipeline, we also execute an ablation study. The study delves into the impact of early termination and uniform scaling of SDFs with or without soft shadows on rendering performance and output quality. By including/excluding these

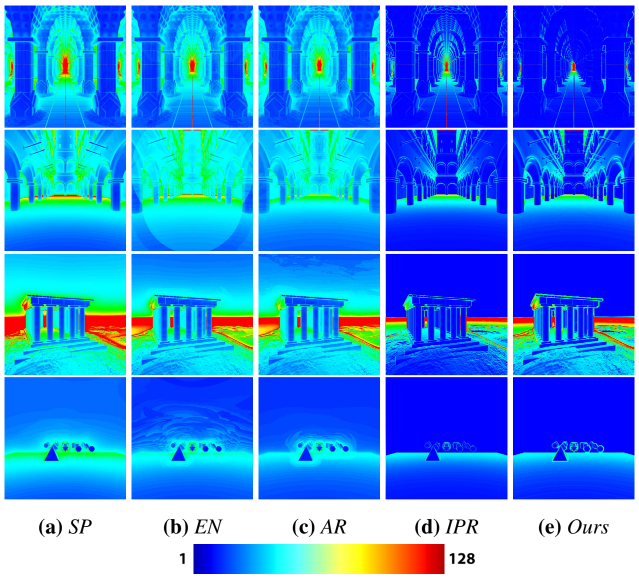


Figure 5: *Left to right, primary rays total step count Sphere Tracing (SP) vs Enhanced (EN) Sphere tracing vs Auto-Relaxed (AR) Sphere tracing vs Inverse Pyramid Rendering (IPR) vs Ours.*

components and analysing their effects, we gain valuable insights into their respective contributions to the overall rendering process.

We leveraged high quality scenes as depicted in Figure 4 (most of them provided by Inigo Quilez, with permission [Ini22]), all represented using procedural SDFs without explicit geometry. The scenes encompass a diverse range of object complexities and environments. The scene "Primitives," serves as an open environment, rendering primitive shapes against a plane with a checkerboard pattern. The scene "Column & Lights" presents a more intricate closed setting created using box SDFs and repeating them using a repetition operator for the floors and the columns.

The scene dubbed "Greek Temple" showcases a landscape, comprising procedurally generated cliff formations and an ancient temple structure. This scene is particularly complex in terms of SDF operators. First, a column is generated by modulating a cylinder SDF using a sine wave function to create the flutes, the vertical grooves running along the shaft of the column. Then the ancient temple involves a repetition operator to generate its multitude of columns. The terrain is generated via a wave function fed by noise sampled from 2D textures. The last scene is the highly complex Sponza atrium (used in [CZR*23]) created by combining various primitive shapes using the subtraction, scaling and repetition operators.

4.2. Performance Evaluation

For the performance evaluation we calculated the time to render a frame and visualise the spawned rays/second for the aforementioned methods with and without shadows, for 1 sample/pixel or 4 samples/pixel Multisample Anti-Aliasing (MSAA) compared to the ground truth (full sphere tracing) which we show in Table 1.

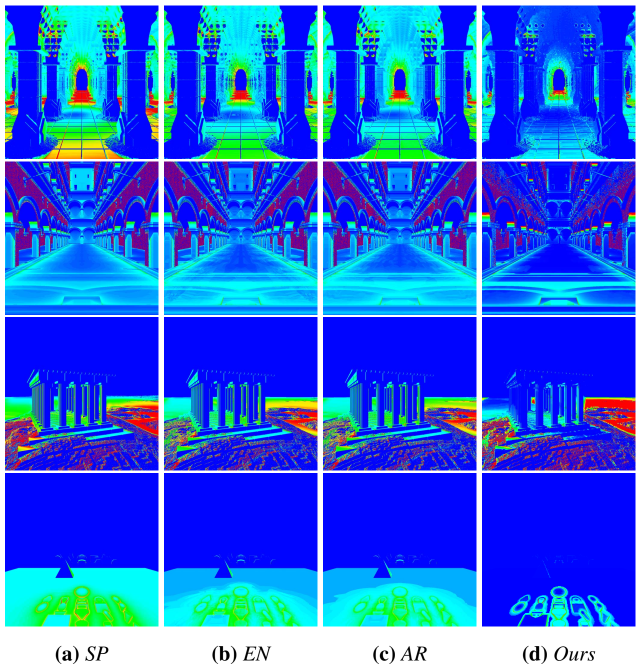


Figure 6: *Left to right, shadow rays total step count Sphere Tracing (SP) vs Enhanced (EN) Sphere tracing vs Auto-Relaxed (AR) Sphere tracing vs Ours. IPR is exluded as it does not support shadow rendering.*

The target resolution was 4K/Ultra HD (3840 x 2160). The computer used had an Intel i9-12900F CPU, 32GB RAM, and a single NVIDIA GPU RTX-3070Ti with 8GB RAM. We present comparison tables of the achieved performance gains against the state-of-the-art and visualize the reduction in steps for all methods.

For the Primitives scene, our method consistently outshines others in terms of frame time (FT) and Million Rays per Second (MR/s), showcasing remarkable reductions in rendering time (speedup 3×) and notably higher efficiency. With anti-aliasing turned off, our method reduces rendering times to half in most scenes compared to the state-of-the-art and with anti-aliasing turned on our method still achieves significant reductions in rendering time. In the same table (1), we also provide a comparison of our method against the state-of-the-art with soft shadows enabled. In the Primitives scene, our method achieved lower rendering time compared to the ground truth with an improvement in performance by 2.1× because most shadow rays finished tracing after 1 step (Figure 6). In the Columns scene, we achieved an improvement of 1.52×, whereas in the Greek Temple scene, we achieved an improvement of 1.8×. In the Sponza atrium, the rendering time increases significantly (shadows take up to 20 ms) compared to other scenes due to scene complexity (increased computations to evaluate more SDFs). However, our method can still improve performance by 1.5×. Overall, our approach achieves better rendering times compared to alternative methods across the board.

Compared to the state-of-the-art, our method outperforms previ-

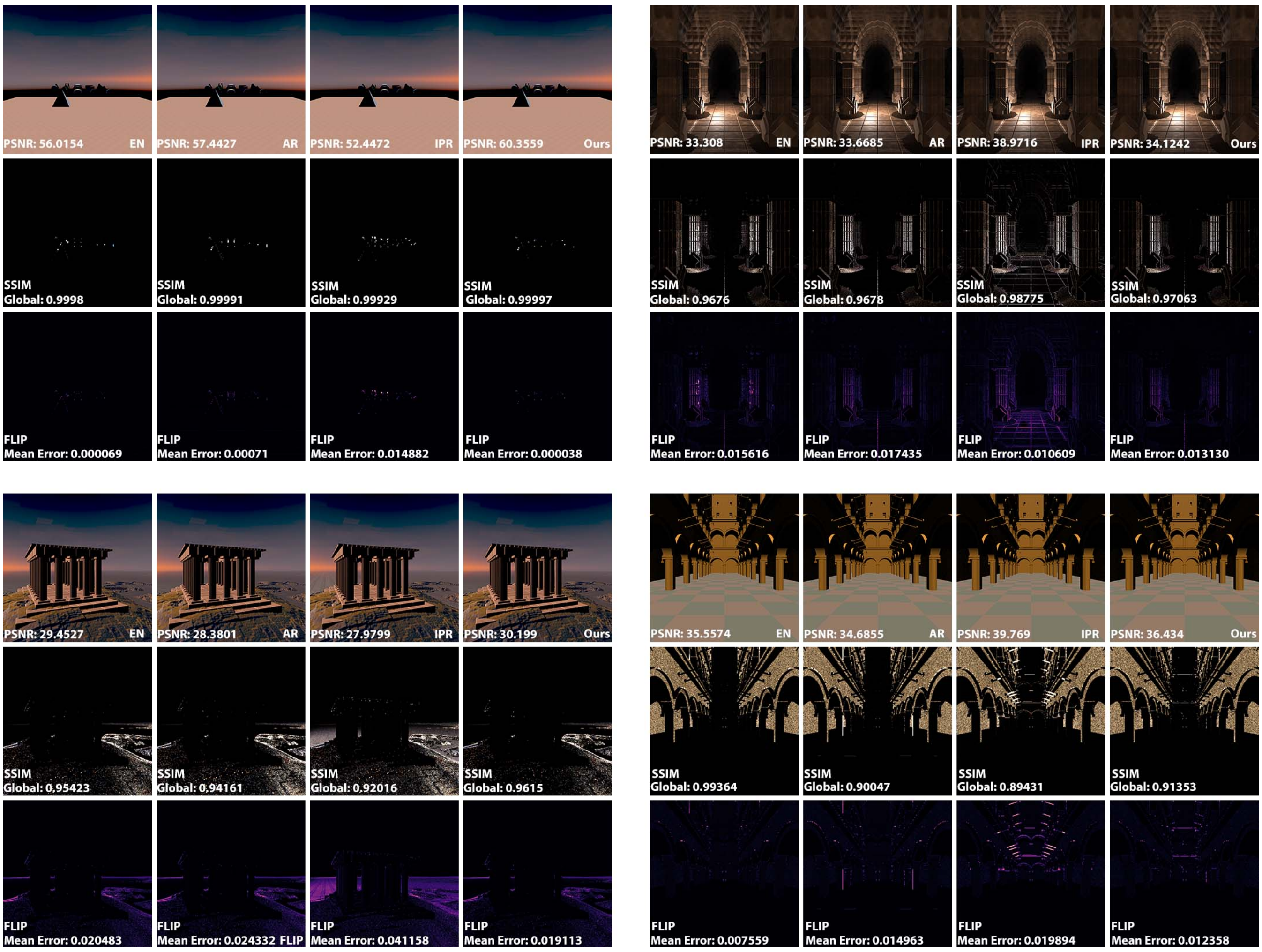


Figure 8: *Visualization of the local SSIM and FLIP difference/error map for the scenes used in evaluation without shadows, compared against ground truth sphere tracing. SSIM images are normalized for better visualization. The global SSIM value indicates high structural similarity and the mean error measured by the FLIP metric approaches zero for all scenes in IPR method and our method, indicating a very high similarity to ground truth with our method outperforming slightly in SSIM and FLIP the IPR and significant in PSNR.*

ous methods as it skips a significant number of steps (Figure 5). For shadow rays, our method improves overall performance by terminating tracing early on the lower resolution buffer based on attenuation resulting in fewer steps overall, and also by reducing the steps in the full resolution rendering by giving an approximate head start to bundles of rays (to 16 rays in the full resolution for just 1 in the low resolution buffer) (Figure 6).

4.3. Visual Quality Evaluation

We employed the Structural Similarity Index (SSIM) [WBSS04], PSNR [HZ10], and FLIP metric [ANA*20] to evaluate the visual quality achieved by our method in comparison to ground truth data obtained via full sphere tracing and state-of-the-art algorithms. Figure 8 provides a visual representation illustrating local SSIM values and the error map produced by FLIP for the aforementioned methods without shadows compared against ground truth. Figure 9 provides a visual representation illustrating local SSIM values and the error map produced by FLIP with shadows. Figures 8 and 9 also present the overall SSIM scores, mean error values resulting from FLIP analysis, and PSNR across different scenes without shadows being rendered for all methods and with shadows for all but IPR (not supported).

For all scenes without shadows, our method achieved higher PSNR, and SSIM & FLIP scores compared to the state of the art, demonstrating that our primary ray sphere tracing produce images closer to the ground truth, i.e., with less artifacts (commonly found around thin structures), than the state-of-the-art. With shadows enabled our method achieves better results compared to the state-of-the-art in two of the four scenes. For the other two scenes our method performs only very slightly worse while being significantly faster. This is not readily reflected in the actual images (Figure 9). Given that we observe a higher error at open environments and only

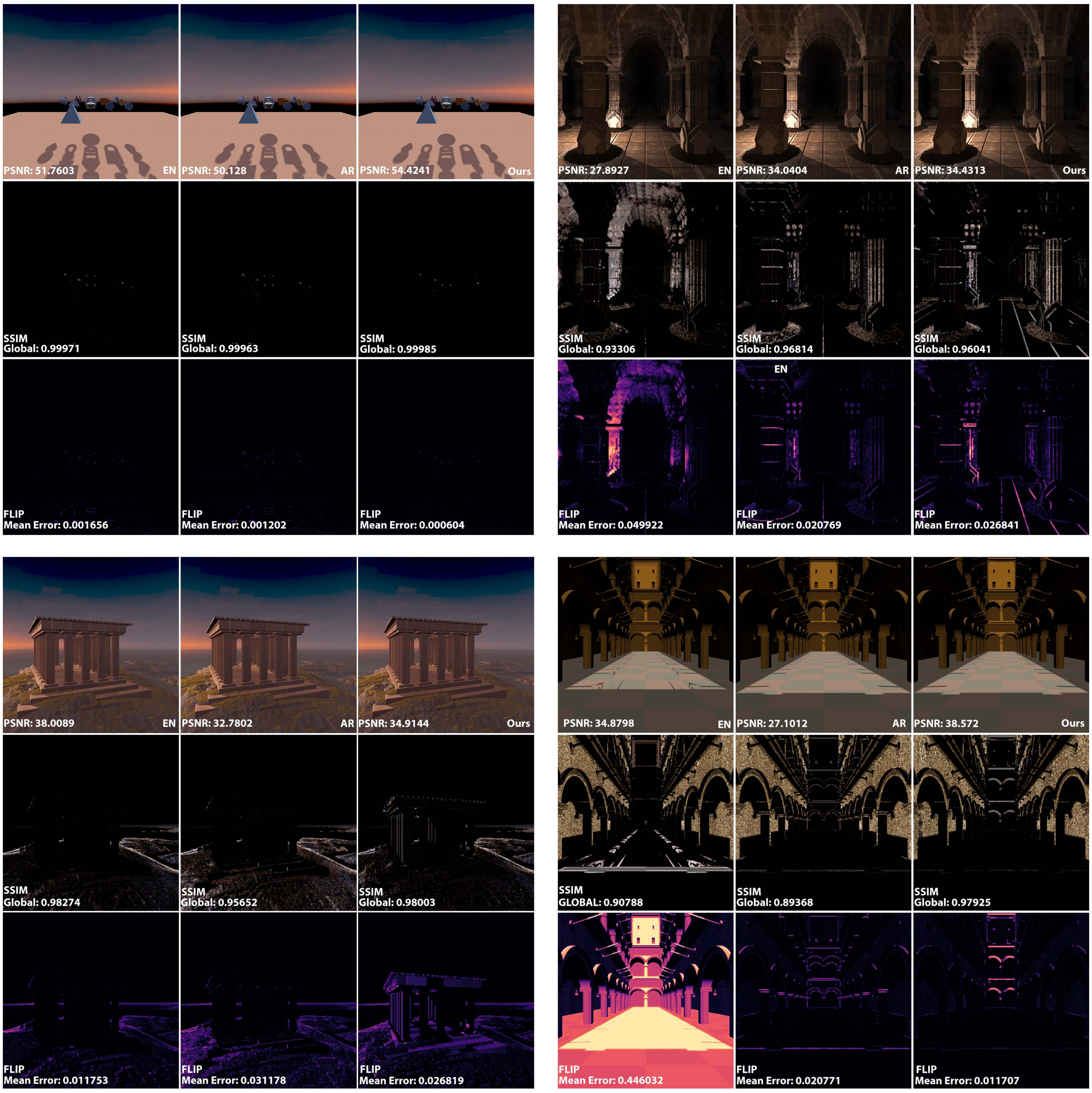


Figure 9: *Visualization of the local SSIM and FLIP difference/error map for the scenes used in evaluation with shadows turned on, compared against ground truth sphere tracing. IPR is excluded from the comparison due to shadows being unsupported. SSIM images are normalized for better visualization. The global SSIM value indicates high structural similarity and the mean error measured by the FLIP metric approaches zero for all scenes in IPR method and our method, indicating a very high similarity to ground truth with our method outperforming slightly in SSIM and FLIP the IPR and significant in PSNR.*

at high-depth values we hypothesize this is due to how we handle depth accuracy in the accelerated low resolution buffer. Our evaluation did not include a perceptual study, as our method produced even less of the thin structure artifacts observed in similar methods (according to PSNR, SSIM & FLIP scores), and those had already been found to be imperceptible [PKM23].

Ablation study. To evaluate the effectiveness of our rendering pipeline's key modules, namely a) early ray termination and b) scal-

	No AA - No Shadows					SSAA x4 - No Shadows				
Scene	SP	EN	AR	IPR	Ours	SP	EN	AR	IPR	Ours
Primitives	16.5	12.7	11.9	10.3	**8.2**	53.9	42.5	58.2	29.9	**14.1**
Columns	7.6	7.5	7.5	6.6	**4.8**	28.4	28.0	26.9	14.8	**14**
Temple	15	12.2	12.01	11	**7**	54.3	43	43	31.2	**24**
Sponza	12.5	11.1	10.1	8.5	**6.4**	45.4	40.1	37.3	25.1	**22.3**

	No AA - Shadows				SSAA x4 - Shadows			
Scene	SP	EN	AR	Ours	SP	EN	AR	Ours
Primitives	20.3	16.0	14.9	**8.7**	52.3	43.2	39.4	**24.9**
Columns	12.3	10.8	10.2	**8.1**	37.2	35.8	34.1	**18.3**
Temple	25.4	18.3	17.9	**14.1**	69.9	51.4	51.2	**39.9**
Sponza	32.8	30.5	28.5	**21.8**	60.5	54.2	52.4	**30.5**

Table 1: *Top sub-table, measurements for primary rays. Bottom sub-table, measurements for primary and shadow rays - excluded IPR from shadow measurements as shadows are not supported. Frame rendering time for each scene in milliseconds; metrics for 1 and 4 samples per pixel. Evaluating the performance of Sphere Tracing (SP), Enhanced Sphere Tracing (EN), Auto-Relaxation Sphere Tracing (AR), Inverse Pyramid Rendering (IPR), and our method for primary and shadow rays. Auto-Relaxed utilized a parameter (b) value of 0.3, Enhanced Sphere Tracing employed an omega value of 0.5.*

		No Shadows			Shadows		
Scene	Metrics	W/O ET	W/O SC	ET & SC	W/O ET	W/O SC	ET & SC
Primitives	FT (ms)	**7.1**	8.4	8.6	**8.3**	10.3	10.5
	PSNR (dB)	44.7198	44.7449	**60.3559**	27.9018	35.2168	**54.3279**
	SSIM	0.99299	0.99402	**0.99997**	0.97347	0.99528	**0.99985**
	FLIP	0.003921	0.003870	**0.000038**	0.031284	0.003200	**0.000604**
Columns	FT (ms)	6.2	**6.0**	6.6	9.0	**8.8**	9.9
	PSNR (dB)	35.4043	37.7992	**38.9716**	24.9847	25.0812	**31.4313**
	SSIM (dB)	0.97911	0.9852	**0.98775**	0.82202	0.82725	**0.96041**
	FLIP (dB)	0.015361	0.011609	**0.010609**	0.011528	0.010869	**0.026841**
Temple	FT (ms)	**7.4**	8.0	8.3	**10.6**	16.4	16.5
	PSNR	25.3528	31.9126	**31.9215**	27.6492	34.1007	**35.168**
	SSIM	0.93149	0.97285	**0.97286**	0.93625	0.97896	**0.97925**
	FLIP	0.039382	0.015598	**0.012358**	0.070749	0.032912	**0.019915**
Sponza	FT (ms)	**6.8**	7.7	8.2	**18.6**	22.1	22.8
	PSNR (dB)	29.0528	35.7455	**39.9769**	27.0548	38.4397	**38.172**
	SSIM (dB)	0.97391	0.98955	**0.99364**	0.89888	0.95915	**0.96332**
	FLIP (dB)	0.024387	0.010603	**0.007359**	0.046430	0.011139	**0.010707**

Table 2: *Results of the Ablation study. Early Termination (ET) and Scaling (SC) enabled or disabled for primary and shadow rays. Both components improve both the computational efficiency and the image quality across the board, with minor exceptions.*

ing of SDFs within the approximated buffer, we conduct an ablation study. The results obtained from this study, with uniform scaling set to 1.005 and ϵ set to 0.05 for the generation of the approximated buffer, are shown in Table 2.

5. Conclusion

We introduced a method for accelerating sphere tracing for rendering SDFs by applying early ray termination and SDF scaling on a low resolution buffer. We extend our method to support high quality accelerated soft shadows, using the same low resolution buffer. We benchmarked the performance of our method compared to ground truth and state-of-the-art methods. Our analysis indicated a significant boost in sphere tracing performance using our method with or without shadows, enabling faster rendering of complex SDF-based scenes without compromising quality. We perform an ablation study providing insights into our method and verifying the important of the integration of the components we propose. Future work may look into a solution to avoid uniform SDF scaling based on depth information obtained during sphere tracing, in addition to dynamic scaling in regions far away from the camera.

Acknowledgment

This paper has received financial support by the Horizon Europe Research & Innovation Programme under Grant agreement N. 101092612 (Social and hUman ceNtered XR - SUN project). Views and opinions expressed in this paper are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the European Commission can be held responsible for them.

References

[Aal18] AALTONEN S.: Gpu-based clay simulation and ray-tracing tech in claybook. In *Game Developers Conference* (2018), vol. 1. 3

[ANA*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 2 (2020), 15:1–15:23. doi:10.1145/3406183. 6

[BBV19] BÁN R., BÁLINT C., VALASEK G.: Area lights in signed distance function scenes. In *Eurographics (Short Papers)* (2019), pp. 85–88. 1, 3

[BK21] BÁLINT C., KIGLICS M.: A geometric method for accelerated sphere tracing of implicit surfaces. *Acta Cybernetica 25*, 2 (2021), 171–185. 3

[BV18] BÁLINT C., VALASEK G.: Accelerating Sphere Tracing. In *EG 2018 - Short Papers* (2018), Diamanti O., Vaxman A., (Eds.), The Eurographics Association. doi:10.2312/egs.20181037. 3, 4

[BV23] BÁN R., VALASEK G.: Automatic Step Size Relaxation in Sphere Tracing. In *Eurographics 2023 - Short Papers* (2023), Babaei V., Skouras M., (Eds.), The Eurographics Association. doi:10.2312/egs.20231014. 3, 4

[CZR*23] CHUBARAU A., ZHAO Y., RAO R., NOWROUZEZAHRAI D., KRY P. G.: Cone-traced supersampling with subpixel edge reconstruction. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–12. doi:10.1109/TVCG.2023.3343166. 1, 3, 4, 5

[Don05] DONNELLY W.: Per-pixel displacement mapping with distance functions. *GPU gems 2*, 22 (2005), 3. 1, 2

[FJL*17] FORSBERG E., JOHNSSON J., LUONG C. T., PERZON A., STRÖMBERG B., ÅBERG J.: Sphere tracing gpu an evaluation of the sphere tracing algorithm and a gpu designed to run it. 1

[FLM24] FERNÁNDEZ-LAYOS P. L.-A., MERCHANTE L. F.: Convex body collision detection using the signed distance function. *Computer-Aided Design* (2024), 103685. 1, 2

[FPSM06] FAYOLLE P.-A., PASKO A., SCHMITT B., MIRENKOV N.: Constructive heterogeneous object modeling using signed approximate real distance functions. 1

[GGPP20] GALIN E., GUÉRIN E., PARIS A., PEYTAVIE A.: Segment Tracing Using Local Lipschitz Bounds. *Computer Graphics Forum* (2020). doi:10.1111/cgf.13951. 1, 3

[GVPN09] GELAS A., VALETTE S., PROST R., NOWINSKI W. L.: Variational implicit surface meshing. *Computers & Graphics 33*, 3 (2009), 312–320. 1

[Har96] HART J. C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (1996), 527–545. 1, 2, 4

[HSK89] HART J. C., SANDIN D. J., KAUFFMAN L. H.: Ray tracing deterministic 3-d fractals. *SIGGRAPH Comput. Graph. 23*, 3 (jul 1989), 289–296. doi:10.1145/74334.74363. 2

[HZ10] HORE A., ZIOU D.: Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition* (2010), IEEE, pp. 2366–2369. 6

[Ini22] INIGO QUILEZ:. https://iquilezles.org/demoscene/, 2022. Accessed: 2022-09-30. 2, 3, 4, 5

[KAS*19] KOULIERIS G. A., AKŞIT K., STENGEL M., MANTIUK R. K., MANIA K., RICHARDT C.: Near-eye display and tracking technologies for virtual and augmented reality. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 493–519. 1

[Kim15] KIM T.: Quaternion julia set shape optimization. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 167–176. 2

[KSK*14] KEINERT B., SCHÄFER H., KORNDÖRFER J., GANSE U., STAMMINGER M.: Enhanced Sphere Tracing. In *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference* (2014), Giachetti A., (Ed.), The Eurographics Association. doi:10.2312/stag.20141233. 1, 3

[Lik08] LIKTOR G.: Ray tracing implicit surfaces on the gpu. In *12th Central European Seminar on Computer Graphics (CESCG 2008 Proceedings). Technische Universität Wien, Institut für Computergraphik und Algorithmen* (2008). 1

[MEG04] MANDELBROT B. B., EVERTSZ C. J., GUTZWILLER M. C.: *Fractals and chaos: the Mandelbrot set and beyond*, vol. 3. Springer, 2004. 2

[MEM*20] MACKLIN M., ERLEBEN K., MÜLLER M., CHENTANEZ N., JESCHKE S., CORSE Z.: Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 3*, 1 (2020), 1–17. 1, 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 2

[NY06] NEWMAN T. S., YI H.: A survey of the marching cubes algorithm. *Computers & Graphics 30*, 5 (2006), 854–879. 2

[OPG21] OECHSLE M., PENG S., GEIGER A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 5589–5599. 2

[PCPMMN21] PUMAROLA A., CORONA E., PONS-MOLL G., MORENO-NOGUER F.: D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 10318–10327. 2

[PFL*13] PARKER S. G., FRIEDRICH H., LUEBKE D., MORLEY K., BIGLER J., HOBEROCK J., MCALLISTER D., ROBISON A., DIETRICH A., HUMPHREYS G., ET AL.: Gpu ray tracing. *Communications of the ACM 56*, 5 (2013), 93–101. 1

[PKM21] POLYCHRONAKIS A., KOULIERIS G. A., MANIA K.: Emulating foveated path tracing. In *Proceedings of the 14th ACM SIGGRAPH Conference on Motion, Interaction and Games* (2021), pp. 1–9. 3

[PKM23] POLYCHRONAKIS A., KOULIERIS G. A., MANIA K.: An Inverted Pyramid Acceleration Structure Guiding Foveated Sphere Tracing for Implicit Surfaces in VR. In *Eurographics Symposium on Rendering* (2023), Ritschel T., Weidlich A., (Eds.), The Eurographics Association. doi:10.2312/sr.20231128. 1, 3, 4, 7

[SEAM22] SÖDERLUND H. H., EVANS A., AKENINE-MÖLLER T.: Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques Vol 11*, 3 (2022). 1, 3

[SN09] SINGH J. M., NARAYANAN P.: Real-time ray tracing of implicit surfaces on the gpu. *IEEE transactions on visualization and computer graphics 16*, 2 (2009), 261–272. 1

[VSJ22] VICINI D., SPEIERER S., JAKOB W.: Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH) 41*, 4 (July 2022), 125:1–125:18. doi:10.1145/3528223.3530139. 1

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612. doi:10.1109/TIP.2003.819861. 6

[Whi79] WHITTED T.: An improved illumination model for shaded display. *SIGGRAPH Comput. Graph. 13*, 2 (aug 1979), 14. URL: https://doi.org/10.1145/965103.807419, doi:10.1145/965103.807419. 1