Interactive Multi-GPU Light Field Path Tracing Using Multi-Source Spatial Reprojection

Erwan Leria
Tampere University
Finland
erwan.leria@tuni.fi

Markku Mäkitalo
Tampere University
Finland
markku.makitalo@tuni.fi

Pekka Jääskeläinen
Tampere University
Finland
pekka.jaaskelainen@tuni.fi

Mårten Sjöström
Mid Sweden University
Sweden
marten.sjostrom@miun.se

Tingting Zhang
Mid Sweden University
Sweden
tingting.zhang@miun.se

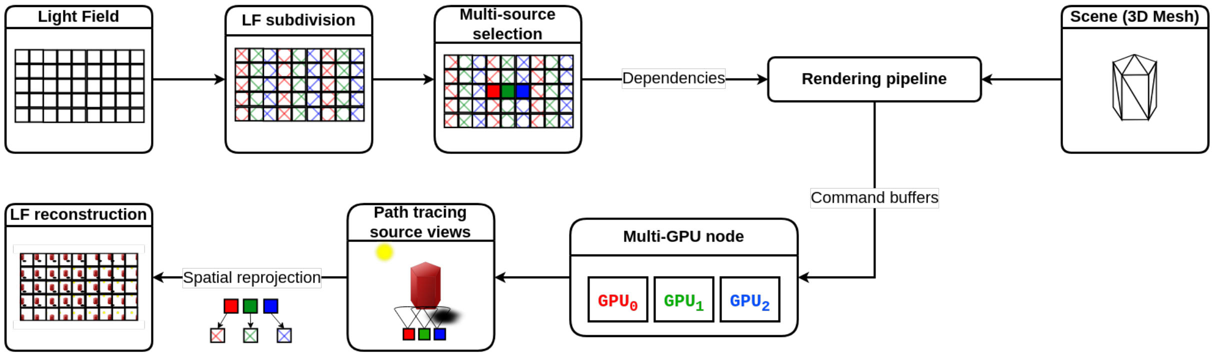


Figure 1: Workflow of our proposed multi-GPU light field path tracing pipeline. A light field is given as input to a light field subdivision algorithm. The light field gets subdivided into as many sets as there are GPUs (in this case three, visualized with red, green, and blue). For each set, we find a local source view that is marked to be path traced, hence ending up with multiple sources equal to the number of GPUs. The other views (targets) will further receive pixels from their local source once it is path traced. These dependencies are passed to the rendering pipeline. The rendering pipeline schedules the command buffer submission to the GPUs. GPUs first path trace the source views. The sources are then used to reconstruct the rest of the light field using spatial reprojection and hole-filling.

ABSTRACT

Path tracing combined with multiview displays enables progress towards achieving ultrarealistic virtual reality. However, multiview displays based on light field technology impose a heavy workload for real-time graphics due to the large number of views to be rendered. In order to achieve low latency performance, computational effort can be reduced by path tracing only some views (source views), and synthesizing the remaining views (target views) through spatial reprojection, which reuses path traced pixels from source views to target views. Deciding the number of source views with respect to the computational resources is not trivial, since spatial reprojection introduces dependencies in the otherwise trivially parallel rendering pipeline and path tracing multiple source views increases the computation time.

In this paper, we demonstrate how to reach near-perfect linear multi-GPU scalability through a coarse-grained distribution of the light field path tracing workload. Our multi-source method path traces a single source view per GPU, which helps decreasing the number of dependencies. Reducing dependencies reduces the overhead of image transfers and G-Buffers rasterization used for spatial reprojection. In a node of 4× RTX A6000 GPUs, given 4 source views, we reach a light field rendering frequency of 3–19 Hz, which corresponds to interactive rate. On four test scenes, we outperform state-of-the-art multi-GPU light field path tracing pipelines, achieving a speedup of 1.65× up to 4.63× for 1D light fields of dimension 100×1, each view having a resolution of 768×432, and 1.51× up to 3.39× for 2D stereo near-eye light fields of size 12×6 (left eye: 6×6 views and right eye: 6×6 views), 1024×1024 per view.

ACM ISBN 979-8-4007-0535-9/24/10
https://doi.org/10.1145/3641825.3687710

ACM Reference Format:
Erwan Leria, Markku Mäkitalo, Pekka Jääskeläinen, Mårten Sjöström, and Tingting Zhang. 2024. Interactive Multi-GPU Light Field Path Tracing Using Multi-Source Spatial Reprojection. In *30th ACM Symposium on Virtual*

Reality Software and Technology (VRST '24), October 9–11, 2024, Trier, Germany. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641825.3687710

1 INTRODUCTION

Physically-based rendering techniques [39], such as path tracing, are ubiquitously used for producing photorealistic computer generated content, since rasterization-based techniques are unable to provide a high level of photorealism without a significant amount of manual effort and pre-computation [10]. Path tracing approximates a solution of the rendering equation [16] through Monte Carlo integration [7]. Due to the stochastic nature of the process, individual samples are highly noisy, and typically thousands of samples (i.e., traced paths) per pixel are needed to reduce the noise to an imperceptible level. However, even with modern high-end GPUs capable of hardware-accelerated ray tracing, only a handful of paths per pixel can be traced in real time on a single GPU. This forces practical rendering applications under real-time constraints to supplement path tracing with complementary techniques, such as noise filtering, and the reuse of existing samples through spatiotemporal reprojection, to increase the quality.

In order to produce immersive and photorealistic 3D content for *light field* (LF) displays, we need to render the scene from dozens or even hundreds of viewpoints. This dramatically increases the amount of computation needed for real-time rendering. Using multiple GPUs becomes necessary to process this amount of data in real-time. On one hand, since the views are typically highly correlated (i.e., they share redundant pixels), computational effort can be reduced by spatially reprojecting existing pixels from path traced views (source view) to target views. However, deciding the number of source views with respect to the computational resources is not trivial: path tracing source views involves a much heavier computational cost than synthesizing views using spatial reprojection. On the other hand, spatial reprojection introduces data dependencies in the pipeline between the views, restricting parallel efficiency. Moreover, not all pixels can be reprojected due to, e.g., view disparity and disocclusions. The disparity is the distance of pixel coordinates of the corresponding points in a source and a target image. Another path tracing stage after the reprojection is required to fill the holes formed by missing pixels. Due to this tradeoff, it becomes essential to group the source and target views onto each available GPU in a way that minimizes the dependencies, the number of missing pixels after reprojection, and the overhead of manipulating image and transfer buffers. The existing literature is missing fundamental methods for selecting multiple sources in a LF for target view synthesis, which motivates our work.

In this paper, we improve upon prior works, addressing deficiencies caused by large data dependencies and fine-grained workload distribution across multiple GPUs for LF path tracing applications that target real-time scenarios, using spatial reprojection for target view synthesis. Our contributions are as follows:

- We demonstrate that trading additional computation against fewer dependencies drastically improves performance and scalability in the LF rendering pipeline on a single node.

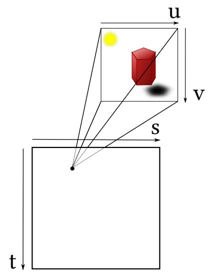


Figure 2: Two-plane parameterization $L(s,t,u,v)$ of a light field. The plane (s,t) contains the view (pinhole camera) sampling positions. Light rays originate from a view on the (s,t) plane and follow a direction sampled on the (u,v) plane. The resulting rendered image is represented on the (u,v) plane.

- We show that assigning tasks to GPUs at a coarse-grained level (on a per-view basis) reduces the amount of dependencies compared to prior works, which decreases the overhead of stages that rely on simultaneous buffer manipulation, such as G-Buffers rasterization and image transfers.
- In the context of multi-GPU LF path tracing, we show our multi-source method for finding multiple source views in view sampling space, which consists in reducing the problem into multiple instances of the Single Source Weber Problem (SWP). In the SWP, the goal is to find the best location for a facility to reduce the overall travel distance to a set of predetermined locations in a Euclidean plane.
- We provide a fully end-to-end pipeline for LF path tracing on a single node, achieving near-perfect linear scalability.

2 BACKGROUND AND RELATED WORK

2.1 Light field displays

For the purpose of this paper, we consider glasses-free multiscopic light field displays and near-eye light field displays [24]. Usually, the term holography is wrongly used as an all-purpose word to describe LF displays. Indeed, holographic displays are based on wave optics, whereas LF displays are based on geometrical optics, considering light as rays. Lately, holographic and LF displays are being investigated and studied in the literature, because they allow to mitigate the vergence-accommodation conflict (VAC) [20], by providing correct depth cues to the eyes [14, 33]. They also provide parallax viewing effect [19, 36, 44].

Glasses-free multiscopic LF displays allow viewers to observe a scene from various angles, thus experiencing a parallax effect. However, these multiscopic displays suffer from a spatio-angular resolution tradeoff. A low spatial resolution restrains the distance at which a viewer can perceive high-frequency details, however it increases the angular range of viewing. Near-eye LF displays [24] further push the boundaries of *Augmented Reality* (AR) and *Virtual Reality* (VR) headsets, releasing the users from visual fatigue and headache caused by the VAC.

LF displays must be given a LF as input in order for a viewer to enjoy their full potential. LFs can be represented by a two-plane parameterization: one plane describes the position of pinhole cameras (i.e., views) from which light rays originate, and the other plane describes the direction of the rays, as shown in Figure 2. LFs can be captured physically with a camera, or synthetically through rendering. Usually, the irradiance of the LF is represented as a flat grid of *elemental images*. Each elemental image corresponds to the irradiance of a view (see Figure 1) for a given resolution.

2.2 Light field reconstruction

LFs are commonly reconstructed from a set of *source views*, i.e., the views rendered (or captured) prior to the reconstruction. They are also called *input* or *reference views* in Image-Based Rendering (IBR).

In real-time graphics, spatial reprojection (or spatial reuse) is a reconstruction technique that is commonly used to save computation time in order to reach low-latency performance, especially in the case of stereo rendering [1, 3, 26, 29, 30, 48]. Spatial reprojection considers rendered pixels in a *source* view and reprojects them to a *target* view, as depicted in Figure 1. This reprojection process is computationally cheaper than tracing light paths for pixels in the target views. The issue of multiview spatial reprojection is that not all the pixels can be reprojected. It is either due to disocclusions or due to the type of reflected surfaces. Non-reprojected pixels produce holes in the target image, so they need to be filled in a separate stage after the reprojection pass.

2.3 Selecting the source views

Deciding the number of source views and their positions is not straightforward and depends on the application and the targeted results. For real-time rendering, Mäkitalo et al. [29] show that path tracing a single source view seems sufficient to achieve real-time performance on a single GPU. Although they show that having more source views reduces the percentage of missing pixels after reprojection, this increases the overall rendering time due to the large overhead of path tracing the additional source views on a single GPU. We tackle the problem in a more general setting, where source views are picked to be path traced and reprojected on multiple GPUs.

Hansen et al. [13] use rasterization to render LFs for a head-mounted display. They only rasterize the four corner viewpoints, and synthesize the rest of the 15×8 grid of viewpoints through reprojection, reaching ~5 frames per second. However, it is difficult to fill in the non-reprojected pixels with their method, whereas with path tracing such cases are easy to handle.

In IBR, especially for LF reconstruction methods based on neural networks, the source views are manually selected [11, 12, 17, 49]. The lack of established method to select the source views leads to unpredictable tendency in terms of computation. For example, in [49], we observe that the proposed neural network is slightly better in terms of computation with 5 source views rather than 9, whereas they show as a comparison that [17] performs better with 9 source views rather than 5.

2.4 Parallel rendering

There are two main parallel rendering schemes for multi-compute devices (i.e., CPUs or GPUs) and/or multi-node rendering: sort-first and sort-last [34]; some methods also employ a hybrid combination of these two schemes.

In image-parallel rendering, the image space is typically subdivided by splitting an image along the horizontal and/or the vertical axis[35], leading to rectangular shaped image subregions. A fine-grained subdivision (i.e., small image subregions) increases the scaling efficiency, but reduces data locality and coherent memory access. This occurs when rays from different subregions or pointing to different objects are run in the same thread group in a GPU. This leads to thread starvation which is caused by *Memory Data Stalling*. Vasiou et al. [41] show this behavior for secondary rays, which exhibit less coherent memory access compared to camera rays.

Antwerpen et al. [4] propose to split the image space into short strips of pixels distributed across the GPUs. They achieve a high scaling efficiency for more than 20 processors, outperforming the tile and scanline subdivisions, for a single view. Xie et al. [50] demonstrate a real-time path tracer in a GPU-cluster for a single view, using 30 samples per pixel and denoising. They perform per-pixel image space subdivision, their method requires using an even number of GPUs per node. Ikkala et al. [15] introduce Tauray, a high-performance real-time image-parallel path tracer that supports rendering stereoscopic and LF content on multiple GPUs in real time. While their framework supports spatio-temporal reprojection and denoising, these stages are executed only on the primary GPU, limiting the full parallelization on multiple GPUs for LF rendering. Leria et al. [26] propose an image-parallel pipeline for real-time stereo path tracing, reusing pixels from one eye to another on multiple GPUs. They improve upon [15] by parallelizing both the reprojection and denoising stages, achieving a rendering frequency of 120 Hz on 4× RTX A6000 GPUs. They do not address the multiview case. Therefore, in their work, the problem of deciding the number and the location of the source views is not addressed.

Rendering large scenes or volumes falls under the sort-last (data-parallel) approach, for which in recent works, the focus has been on improving data space partitioning [52] of the scene and/or the scheduling of rays through the different data subdomains [38, 43].

In this work, we establish our scope of study with the image-parallel approach. This approach allows us to get a control over the distribution of the views to the GPUs. Many previous works render multiview images using distributed and parallel computational resources [9, 21, 22, 27, 46]. However, they do not address the implications between dependencies and scalability caused by the multiview reconstruction process.

3 MULTI-SOURCE SPATIAL REPROJECTION

Spatial reprojection involves additional data dependency to the otherwise parallel path tracing pipeline from a source pixel to its target point. Previous works in real-time graphics consider only a single source view, in order to minimize the initial path tracing overhead, but in the process worsen the critical path of computation. In our approach, we improve data locality and data dependency by subdividing the LF into sets, each having its independently path traced source view. This way, each set is assigned to a separate GPU, enabling fully independent execution at the cost of path tracing overhead due to additional path traced source views. In this section, we explain our approach for selecting multiple source views from a LF.

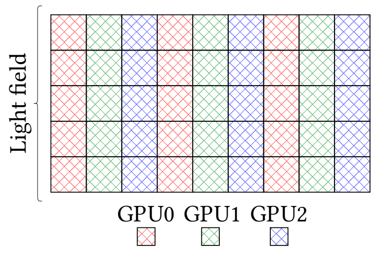


Figure 3: Illustration of a light field subdivided into three sets (red, green, and blue). The views (squared cells) are assigned to a set in a round-robin fashion. Each set is then associated with a GPU.

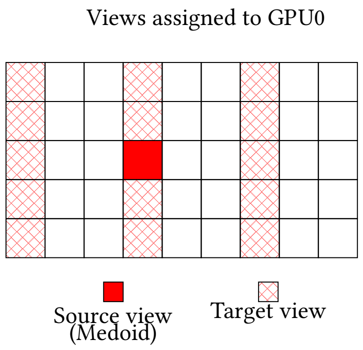


Figure 4: Illustration of the position of the medoid in the red set. The medoid (source view) is represented by a squared cell filled with solid red color. The target views are represented with squared cells filled with a crosshatch pattern.

3.1 Light field subdivision

We identify two key problems in subdividing a light field: Deciding the number of source views, and building dependencies from a source to some targets. To consider the number of sources, which are fully path traced, one must consider the available computing resources. In our pipeline, we assume a single node composed of N GPUs, and we want to render M views. When denoising subregions of a view split to multiple GPUs, seam artifacts appear between stitched subregions [26]. Thus, we do not consider the case where $N > M$, which would force each view to be split into multiple subregions of pixels.

We group views into disjoint sets of views. We decide that each set has a single source view. Each set is then later assigned to a GPU. It is up to the implementation to decide the strategy for assigning some views into a set. Once a set is assigned to a GPU, we then begin to look for the source.

Unlike prior works, we do not arbitrarily select the sources in the LF plane. We automatize the selection process. Since we are searching for a single source within a set, we would like to select the most central view within the set. As explained by Okutomi et al. [37], when two views are spatially close to each other, the range of disparity is minimized. Thus, a lower disparity allows to reproject more pixels. This is shown for different LF configurations in [29]. We define the central view as the view that is spatially minimizing the distance to all the other views in its set. When forming a polygon with the coordinates of the views from the same set, the polygon may have a non-convex shape in the LF plane. This depends on the assignment algorithm. Therefore, for a set, computing the arithmetic median or the arithmetic mean from the coordinates of the views fails to provide sufficient accuracy to determine the central view. Indeed, the given point could lie outside the boundaries of the non-convex shape. Instead, the optimal solution is to identify the *medoid*.

Finding sets of views and their corresponding medoids in a single algorithm can be achieved through the Partitioning Around Medoid (PAM) algorithm [18], where the medoid of each partition is recomputed at each insertion of a new element in its partition. However, PAM does not guarantee that views are assigned as evenly as possible, which in our approach, is handled by an assignment algorithm that does not compute the medoid, allowing different assignment strategies to control the workload distribution. Using a method such as PAM does not allow for this degree of freedom.

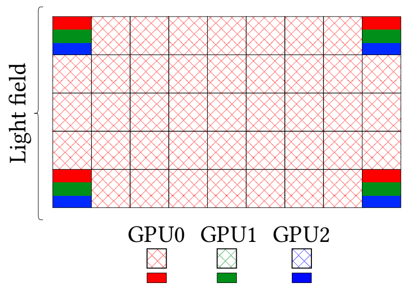
In a set of views, finding a single source view that minimizes the Euclidean distances to all the other views in a plane is an instance of the Single source Weber Problem (SWP) [45]. In each set, the medoid is approximated by selecting the closest viewpoint to the *geometric median*. We employ the Weiszfeld algorithm [47] to find the geometric median of each set; this algorithm is commonly used to solve the SWP [8]. When each medoid is found, it becomes the source view for its respective set, and the remaining views of the set become the target views.

3.2 Multiview rendering dependency

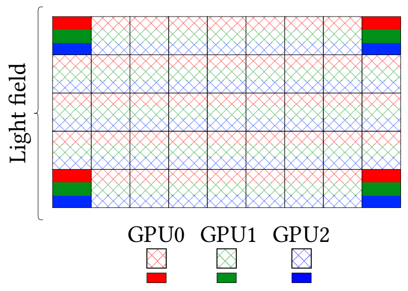
We want to reduce the dependencies caused by spatial reprojection in order to reduce buffer read/write overhead. The number of buffers per GPU depends on the number of source and target views that are used for reprojection.

We break down the combinatorial complexity of dependence when using spatial reprojection and multiple views and multiple GPUs as follows. Given a two plane parameterization $L(s, t, u, v)$ of a LF, consider two views A and B located on the (s, t) plane. Let p_a be some pixel in view A and p_b be some pixel in view B, both located on the (u, v) plane. When spatial reprojection is called between pixels p_a and p_b, this implies that p_b depends on p_a that we denote as $\delta(p_a, p_b)$, or more simply $\delta(A, B)$ since reprojection is applied from one view to another. This means that A contains the source data of B. This implies that the pixels in A and the ones in B can not be computed in parallel because of their dependency.

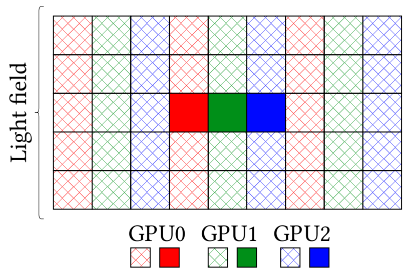
Note that if there are M views and k sources, there can be at most $k \cdot (M - k)$ δ-dependencies between views. If N GPUs are working on one n^{th} of each view, then, the number of dependencies goes up to $N \cdot k \cdot (M - k)$. The implementation of the spatial reprojection algorithm decides the conditions for whether to reproject a pixel or not. This means that for each dependency $\delta(a, b)$, reprojection may not occur, but the dependencies remain until the feasibility of reprojection is determined. In our approach, we encapsulate inter-view dependencies within a set of views. We avoid spreading dependencies from all source views to all target views, as opposed to the work done in [15, 26]. This way, each view is confined within a single set and thus to a single GPU.



(a) Example of the light field distribution used in [15].

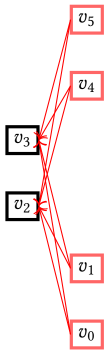


(b) Example of the fine-grained light field distribution used in [26].

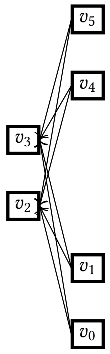


(c) Example of the coarse-grained light field distribution used in our proposed method.

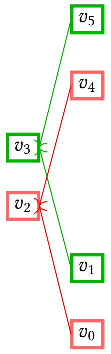
Figure 5: We illustrate three different light field workload distribution schemes. The colors depict the data locality of the views, each color corresponding to a GPU (red, green, and blue). Solid colors depict the source views, colored crosshatched pattern depicts the target views. When a view is split across the 3 GPUs, it is divided into three regions. a) On the left, workload distribution scheme as used in [15], resulting in a fine-grained distribution of the path traced source views. The target views are handled on the primary GPU (red - GPU$_0$). b) In the middle, light field subdivision based on a fine-grained distribution of source views and target views to the GPUs [26]. In (a) and (b), 4 source views (corner views) are path traced. c) On the right, we show the proposed coarse-grained distribution scheme.



(a) Ikkala et al. [15]
$D = k \cdot (M - k)$



(b) Leria et al. [26]
$D = k \cdot (M - k) \cdot N$



(c) Proposed
$m = \frac{M}{N}, D = M - N$

Figure 6: Inter-view data dependency graph during the spatial reprojection stage between 2 source views and 4 target views with 2 GPUs (M = 6, N = 2, k = 2). The views v_2 and v_3 are source views. The target views are v_0, v_1, v_4 and v_5. The color encoding illustrates the locality of the views (squared nodes) and the dependencies of the targets (arcs): red for GPU$_0$, green for GPU$_1$, black for both – similarly to the source views in Figure 5a and Figure 5b. Consider that black arcs and squared cells are equivalent to red and green colors being overlapped. The number of dependencies D is given for each pipeline.

In our pipeline, the spatial reprojection stage has at most $k_s \cdot (m - k_s)$ dependencies per GPU, with k_s the number of sources per set (see Figure 5c). We choose the number of sources to be equal to the number of GPUs, that is, $k = N$, so 1 source per set. Indeed, when N converges towards M, we want to reach linear scalability, rendering only a single view per GPU when $N = M$. We define the size of a set m to be equal to one n^{th} of the entire LF, which gives the following number of dependencies between source and target views: $D = (k_s \cdot (m - k_s)) \cdot N = (\frac{M}{N} - 1) \cdot N = \mathbf{M - N}$, with $m = M/N$ and $k_s = 1$. As shown in Figure 6, when $k = N$, our multi-source method shows fewer dependencies compared to the other pipelines.

3.3 Pipeline

On each GPU, our rendering pipeline has four main sections: path tracing, LF reconstruction, post-processing and transfers. In the first section, each GPU path traces a single source view. In the second section, GPUs reproject the path traced pixels from the source view to some target views, and then use path tracing to fill the non-reprojected pixels. In the third section, algorithms such as denoising or tonemapping are applied. Finally, all the views are sent to the primary GPU.

In order to discard pixels that are not reprojectable, the spatial reprojection algorithm uses information from the G-Buffers (normals, material, world positions). Denoising also uses information provided by the current G-Buffers, and by the G-Buffers from the previous frame as well. In our pipeline, a GPU rasterizes the G-Buffers for each view it is associated to. The dependency pattern is pipeline dependent (see Figure 6), and therefore known before the command buffers are submitted to the GPUs. This allows us to schedule computation at each stage of the pipeline based on the views to process. Determining the view indices of the sources and the targets depends on the assignment algorithm. These view indices and the dependencies are used to schedule computation.

4 EVALUATION

We implemented our solution into Tauray [15], an open source stereo and LF real-time path tracer based on Vulkan. Our proposed multi-source method for multi-GPU LF path tracing is evaluated on the basis of runtime performance and scalability benefits.

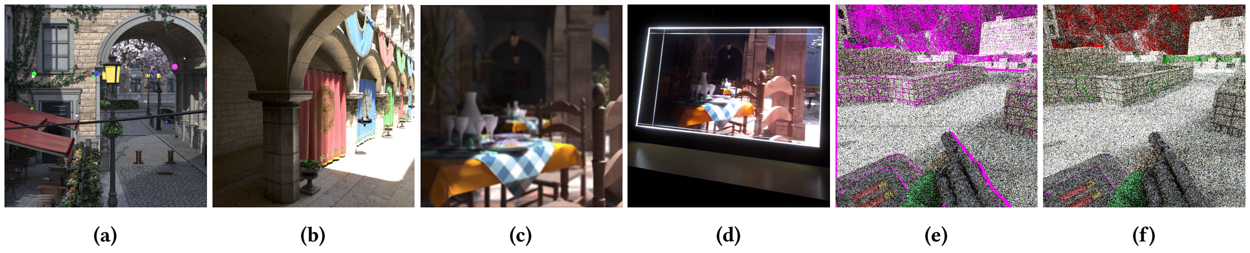


Figure 7: Example views of our 4 test scenes. a) 4096 spp Bistro Exterior [28]. b) 4096 spp Sponza © Crytek [32]. c) 1 spp denoised retinal image of San Miguel © Guillermo M. Leal Llaguno [32]: the retinal image of the left eye, using the NELF display model described in [12]. The focus of the eye is set at 35 cm: the chair in the foreground appears sharp, the rest of the image is affected by the defocus blur. d) Example of our Multiscopic setup with San Miguel 1 spp denoised on the Looking Glass Display (LKG) 32". e) 1 spp Eternal Valley FPS [2, 51], with our pipeline: the image shows pixels reprojected by a single GPU and from a single source, the missing pixels are marked with the dark pink color, f) same image for the pipeline described in [26]: 4 GPUs are reprojecting pixels in the same view, the missing pixels are encoded with a color corresponding to their GPU region (red, green, blue, dark pink). The exposure of the images has been increased for better visibility. The resolution of the views is 1024×1024, except for (d), where the views displayed have a resolution of 768×432 each.

4.1 Experimental setup

Our computing platform for the experiments is a single node composed of 4× RTX A6000 48 GB GPUs on a PCIe 4.0 16GT/s. Our platform does not have NVLink, which would further reduce the latency for GPU-to-GPU transfers, reaching a peak bandwidth of 20 GB/s up to 40 GB/s. For path tracing, we use 1 spp and a maximum ray depth of 6. Our spatial reprojection algorithm reprojects only diffuse surfaces. We denoise each viewport using Spatiotemporal Variance-Guided Filtering [40].

4.1.1 Scenes. We are using four scenes (see Figure 7): Bistro Exterior ($13M$ triangles), Sponza ($262k$ triangles), San Miguel ($4.9M$ triangles), Eternal Valley FPS ($251k$ triangles). We modified the Bistro Exterior scene such that it contains more triangles to increase the workload during G-Buffers rasterization: we replaced the tree in the courtyard by a tree composed of ~10M triangles, mostly located in the leaves. The Bistro Exterior scene has slightly moving objects. The Eternal Valley FPS scene contains skeletal animations for multiple characters, and it has dynamic *First-Person Shooter*-like camera movements, with the total animation length of the camera movement being 360 frames. For this reason, we set up the runs for all scenes to last for 360 frames.

4.1.2 Light field setup. We choose two different configurations for the LFs that we render for the experiments; the configurations correspond to practical LF displays:

***Multiscopic**. Multiscopic* will refer to a 1D LF composed of 100 views that are arranged for horizontal parallax effect. Considering a display panel of resolution 7680×4320 (8K), we divide it into 100 views, each having a resolution of 768×432. Figure 7d shows an example of the San Miguel scene displayed on a multiscopic display.

***Near-Eye Light Field** (NELF).* NELF will refer to a 2D LF composed of 72 views, with 6×6 views for each eye. The 6×6 blocks are separated by a distance of 6.2 centimeters, corresponding to the interpupillary distance. Each view has a resolution 1024×1024. This resolution corresponds to the resolution of some liquid crystal on silicon spatial light modulators (SLMs). SLMs are used for microdisplay systems, such as near-eye displays or microscopes, due to their high speed for updating the light intensity and their compact form factor [25].

In the two LF setups, neighboring views are spaced by 1 millimeter because we want to keep a low disparity to benefit from reprojection and avoid the views at extremities to end up outside the scene boundaries.

4.1.3 Comparison baselines. Our comparison baselines are two works from the literature, both based on Tauray [15, 26]. The work done in [26] addresses the stereo case, we extended it, such that it suits the multiview case. For our solution, we use a round-robin assignment to distribute the views to their set. Finding the source views is not automatized for the baselines, they are selected manually: when the number of source views is odd, the sources are in the center of the LF, otherwise they are in the corners.

To prevent any confusion, we use the term rendering frequency (Hz) to indicate the frequency at which a LF is rendered, and we use elemental images per second (ei/s) to describe the number of elemental images rendered in one second.

4.2 Pipeline stages

The timing results for each stage of the proposed pipeline are given in Table 1. Stages are enumerated by order of command buffer submission to the GPUs, although some stages may overlap, especially stages 8 and 9: the current G-Buffers are recorded for the next frame, during transfers. For the hole-filling stage (5), we observe that [26] has a better rendering time than our proposed solution by a factor ×1.09, this is due to the fact that more sources benefit more targets. However, the counterpart is that our proposed solution has much fewer dependencies, involving fewer buffers, which is showcased by the improved execution time of the G-Buffers rasterization stage (3) and the transfers to the primary GPU (9), respectively, by a factor ×4.11 and around ×1.8. In the pipeline of [15], only the source views are path traced and transferred to the primary GPU, while in [26] and in our proposed solution, sources and targets are both transferred. As a general indication, for the NELF setup, G-Buffers

	Execution time (ms)		
Pipeline stages	**[15]**	**[26]**	**Proposed**
0. Skinning	0.01	0.08	0.038
1. Scene update	0.3	0.08	0.47
2. Path tracing	65.01∗	9.28	9.84
3. G-Buffers rasterization	166.67∗	147.36	35.85
4. Spatial reprojection	1.44∗	1.42	0.91
5. Hole-filling	355.06∗	69.62	75.67
6. Denoising	67.32∗	14.32	17.50
7. Tonemapping	0.07*	0.39	0.32
8. Record previous buffer	7.65∗	21.70	20.65
9. Transfer to primary GPU	1.43 + 0.21	21.45 + 5.03	13.18+1.27
10. Stitching image regions	0.09*	n/a	n/a
11. Blit images to the swapchain	n/a	3.16∗	2.72*

Table 1: Averaged timings per stage in milliseconds for Bistro Exterior over 360 frames with the Multiscopic setup. The transfers show first the timing to send, plus the timing to receive data. Recording previous buffers (8) occurs during the transfer of images (9). The asterisk (*) indicates stages that are run only on the primary GPU, which is defined by the design of the pipeline.

rasterization accounts for 14% up to 27% of the rendering time, depending on the scene. The hole-filling stage accounts for 13% up to 51%.

4.3 Execution time

We provide an overview of the average rendering time per frame with the 4 test scenes in Table 2. We see that our proposed solution consistently outperforms the baselines. For the Multiscopic setup, we report a speedup of ×1.66 up to ×4.63 and for the NELF setup, a speedup of ×1.51 up to ×3.39. We notice that the size of the scene has an impact on the overall computing time.

In Table 3, we show the performance of the hole-filling stage. For the Multiscopic setup, we notice that our pipeline does not beat the pipeline of [26] on the 4 test scenes. However, it remains acceptable compared to [15]. This difference in terms of performance is more noticeable for the NELF setup. The dependencies caused by spatial reprojection in [26] implies that k sources reproject their pixels to $M - k$ targets and thus reduce the number of missing pixels, whereas in our pipeline, only a single source reprojects its pixels to M/N targets.

We show the reconstruction performance per GPU in Table 4. First, we notice that the ratio of missing pixels between the views is uniformly distributed among the GPUs. However, the uniformity of the execution time of the hole-filling stage per GPU is less stable for the Multiscopic setup. On the other hand, we notice that for the NELF setup, the execution time of hole-filling stage is more even. This is due to the views being more compact in the NELF setup, compared to the Multiscopic setup, where the distance between two views can be higher than in the 2D setup. The stability of the execution time of the hole-filling stage for the NELF setup seems to also be due to the lower ratio of missing pixels compared to the Multiscopic setup.

4.4 GPU utilization

Figure 8 shows the GPU utilization of 4 GPUs for 4 source views. In Figure 8(a), we observe for the pipeline of [15] that the GPU utilization of the non-primary GPUs is very low. From the collected data, we estimate that the average percentage of underutilization is 63%. For Figure 8(b), we can see that for the pipeline of [26] the utilization per GPU looks to be mostly above 50%; the average percentage of underutilization is around 15%. For our proposed solution (Figure 8(c)), we see that the GPU utilization is very high compared to the baselines. The average percentage of underutilization is under 1%, which shows that a constant stream of work is provided to the GPUs with little idle time.

4.5 Scaling

Figure 9 demonstrates the scalability of our proposed solution against the baselines for the Multiscopic (a) and the NELF (b) setups. We observe first that the idea of trading more computation time to render more sources is efficient for the two baselines. However, in [15], the primary GPU has to take on the tasks of G-Buffers rasterization, spatial reprojection, hole-filling and denoising alone, so it ends up being overloaded. On the other hand, [26] seems to scale slightly better, but due to its dependencies it does not scale linearly. For example, we observe in the Multiscopic setup with 2 sources, we outperform the baselines with only 2 GPUs, whereas they are both using 4 GPUs, getting a roughly ×1.77 speedup. Overall, for the two LF setups, we see that our approach to reduce dependencies tends towards near-perfect linear scalability.

In Figure 10, we show the performance of our implementation in terms of scalability when the number of GPUs is fixed and the number of views increase. For this experiment, the width of a LF increases from 1×10, up to 10×10. Until 3×10 views, the difference between 2 and 4 GPUs does not appear to be significant, since the weight of the workload is not important. When the number of views becomes larger ($> 3 \times 10$) the rendering pipeline benefits from a higher number of GPUs since the workload is heavier. By observing Figure 9 and Figure 10, with respect to Table 2, when increasing the number of GPUs, we notice that the performance gap is widening more rapidly for complex scenes (e.g., Eternal Valley FPS) compared to simpler static scenes (e.g., Sponza).

4.6 Assignment variation

In Table 5, we demonstrate that our solution is, at least, agnostic to any naïve assignment algorithms for grouping views by set. We compare the one-to-one Round Robin assignment against the Contiguous assignment, with each contiguous block containing M/N elements. Regardless of whether a LF is 1D or 2D, we consider the index of the views to be a one-dimensional array. Then we apply the assignment algorithms on this index array. Although it is not in the scope of this paper, we observe that the assignment method to assign views to sets has an impact on the performance. We see that Round Robin has a speedup factor of ×1.76 for the NELF setup compared to Contiguous assignment.

4.7 Discussion

Overall, our results show that increasing the number of source views, at least up to the number of GPUs, improves the overall

	Average rendering time per frame (ms)							
Scene	Sponza		Eternal Valley FPS		Bistro exterior		San Miguel	
LF setup	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF
[15]	154.8	369.3	395.5	525.8	813.1	1081	353.1	516.8
[26]	87.4	242.8	347.3	414.3	370.2	481.2	280.0	323.3
Proposed	**52.7**	**123.6**	**85.5**	**155.1**	**211.3**	**318.8**	**110.2**	**173.1**

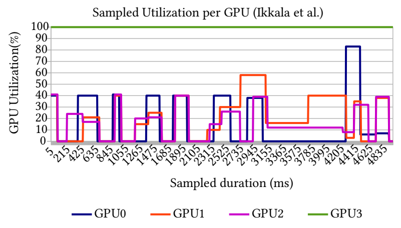
Table 2: Rendering time per frame using 4 GPUs averaged over 360 frames, for each scene.

	Hole-filling (ms)							
Scene	Sponza		Eternal Valley FPS		Bistro exterior		San Miguel	
LF setup	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF
[15]	20.76	26.92	21.35	20.09	355.06	571.13	170.24	175.61
[26]	3.6	7.03	4.21	6.85	69.62	123.91	43.57	40.12
Proposed	8.81	16.43	9.33	20.65	75.67	165.11	52.84	70.77

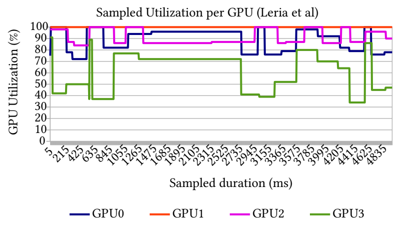
Table 3: Hole-filling performance per scene and per LF setup. We use 4 GPUs and 4 source views. Timings are averaged over 360 frames

	Reconstruction performance per GPU							
	GPU 0		GPU 1		GPU 2		GPU 3	
LF setup	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF	Multiscopic	NELF
Spatial reprojection (ms)	0.91	1.99	0.91	2.00	0.91	2.99	0.91	2.06
Ratio of missing pixels (%)	48.43	35.74	48.45	35.75	48.47	36.17	48.48	35.92
Hole-filling time (ms)	80.67	165.37	73.73	161.44	73.85	164.93	74.45	168.72

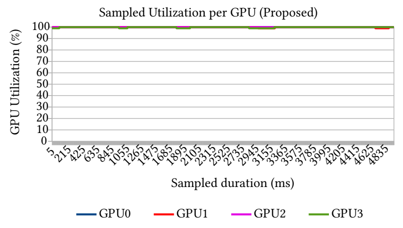
Table 4: Reconstruction performance results of our pipeline per GPU for the Bistro Exterior scene, averaged over 360 frames. The reconstruction timings are shown for 4 GPUs and 4 source views.



(a) Ikkala et al. [15]



(b) Leria et al. [26]



(c) Proposed

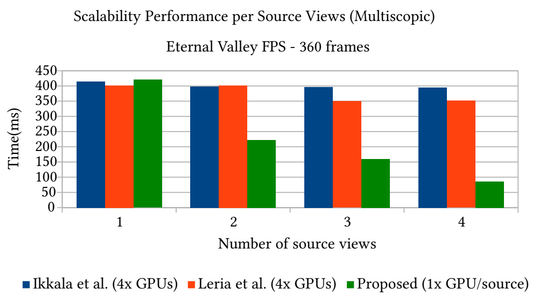
Figure 8: GPU utilization extracted from a run of 360 frames of the San Miguel scene, static camera, NELF setup. We use 4 GPUs and 4 source views. The GPU utilization is sampled with *nvidia-smi* over a duration of 5000 ms. The GPU utilization is recorded every 5 ms.

Assignment method	Round Robin		Contiguous	
LF setup	Multiscopic	NELF	Multiscopic	NELF
Average rendering time per frame (ms)	111	173	120	305

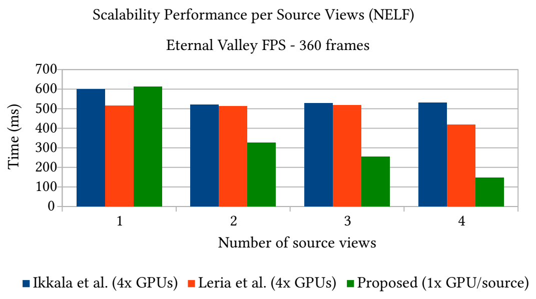
Table 5: Performance of Round Robin and Contiguous assignment algorithms for the San Miguel scene for a run of 360 frames.

performance of the proposed pipeline, but also of the baselines used for comparison. When averaging the rendering time per frame for each scene, we have a speedup of ×3.26 against [26].

We observe in general that there are fewer missing pixels after the reprojection stage in 2D LFs than in 1D LFs. This is due to the larger number of adjacent views in 2D. However, in terms of rendering time, 2D LFs are more demanding. In fact, there are fewer views in the NELF setup compared to the Multiscopic setup, but the total amount of pixels in the NELF setup is nearly two times larger compared to the Multiscopic setup. Some NELF displays are using time-multiplexing techniques to counter the issue of limited spatial bandwidth in SLMs [6], with the intent to reach the target critical flicker frequency (CFF) of the human vision of 90 Hz [31]. However, in this work, we notice that reaching this CFF for LF displays is currently out of reach for multi-GPU LF path tracing, using spatial reprojection and with a high number of views. This demonstrates a discrepancy between the advancement of multiview



(a) Multiscopic



(b) NELF

Figure 9: Scalability performance per number of source views. The rendering time per frame (in ms) on the y-axis is averaged over 360 frames of the Eternal Valley FPS scene. Results are shown for the Multiscopic (a) and the NELF (b) setups. Ikkala et al. [15] and Leria et al. [26] are measured with 4 constant GPUs ($N = 4$). Our proposed solution is measured with 1 GPU per source ($N = k$).

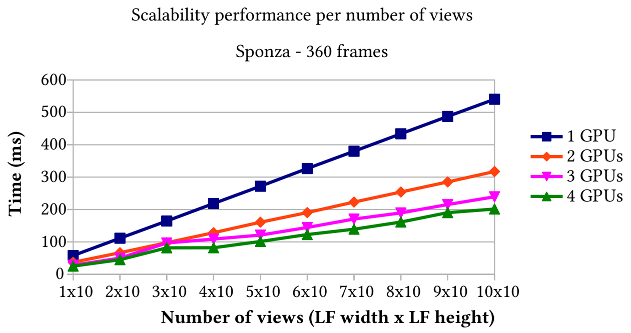


Figure 10: Scalability performance per number of views with the Sponza scene. The resolution of the LF is indicated on the x-axis, each view has a 1024×1024 resolution. The average rendering time per frame is indicated on the y-axis in milliseconds. The rendering time is shown for 1 up to 4 GPUs. The number of source views is equal to the number of GPUs.

displays and the actual state of graphics hardware and techniques to render photorealistic images at high frequency.

Considering mobility for future AR/VR applications, traditional wireless stereo AR/VR spatial computing devices offer a more practical solution for scenarios with lower bandwidth requirements. In contrast, NELF displays would require a wireless network bandwidth of approximately 20.38 GB/s to transfer an uncompressed $12 \times 6 \times 1024 \times 1024$ RGB-8bits LF from a remote server at 90 Hz, which is highly demanding. On a Wi-Fi 8 network [23] with a bandwidth of 100 Gbps (12.5 GB/s), this would require roughly a 2:1 ratio of compression, which is easily achievable via standard low-complexity codecs [42]. However, decoding the images on a single client GPU is a bottleneck for low-latency streaming when the number of pixels is large (4k resolution and more), as shown in [5].

This work shows highly promising results, highlighting the benefits of scalable performance in multi-GPU LF path tracing pipelines. For the NELF setup, we reach a frequency of 3 Hz up to 8 Hz depending on the scene, respectively 226 ei/s and 582 ei/s. On the other hand, for the Multiscopic setup, we reach an interactive rendering frequency of at least ~4 Hz for heavy scenes like Bistro Exterior, up to ~19 Hz for smaller scenes like Sponza. Respectively, these correspond to 474 ei/s and 1897 ei/s. All setups combined, the comparison baselines reach a rendering frequency of 0.9 Hz up to 11 Hz, and equivalently 66 ei/s up to 1144 ei/s.

5 CONCLUSION

In this paper, we demonstrated an important computational complexity-to-scalability tradeoff in sample reprojection enhanced multiview path tracing, with the aim of targeting LF display scenarios for horizontal parallax and VAC-free immersive VR. Our multi-source spatial reprojection approach is agnostic to any assignment algorithms. The proposed method restrains the reuse of pixels based on coarse-grain work assignment, as opposed to prior works. This reduces data dependencies, decreasing the overheads caused by G-Buffers rasterization and image transfers to the primary GPU.

The implementation of our pipeline outperforms prior works, showing a significant improvement in the overall rendering time (1.51–4.63× speedup on 4 GPUs) and in GPU utilization, reaching over 99% utilization on 4 GPUs. This way, the renderer can reach near-perfect linear scalability, achieving interactive LF rendering frequency of 3–19 Hz.

ACKNOWLEDGMENTS

Thanks to Ugur Gudelek for sharing and parameterizing the NELF display model. We would like to thank the Centre for Immersive Visual Technologies of Tampere University for giving us the opportunity to use the LKG 32" display. This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 956770.

REFERENCES

[1] S.J. Adelson and L.F. Hodges. 1995. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications* 15, 3 (1995).

[2] Joel Alanko, Markku Mäkitalo, and Pekka Jääskeläinen. 2022. TauBench: Dynamic benchmark for graphics rendering. In *INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS THEORY AND APPLICATIONS*.

[3] Magnus Andersson, Björn Johnsson, Jacob Munkberg, Petrik Clarberg, Jon Hasselgren, and Tomas Akenine-Möller. 2011. Efficient multi-view ray tracing using edge detection and shader reuse. *The Visual Computer* 27 (2011).

[4] Dietger van Antwerpen, Daniel Seibert, and Alexander Keller. 2019. A simple load-balancing scheme with high scaling efficiency. In *Ray Tracing Gems*. https://www.realtimerendering.com/raytracinggems

[5] Tim Biedert, Peter Messmer, Thomas Fogal, and Christoph Garth. 2018. Hardware-Accelerated Multi-Tile Streaming for Realtime Remote Visualization. In *Eurographics Symposium on Parallel Graphics and Visualization*, Hank Childs and Fernando Cucchietti (Eds.).

[6] Suyeon Choi, Manu Gopakumar, Yifan Peng, Jonghyun Kim, Matthew O'Toole, and Gordon Wetzstein. 2022. Time-multiplexed Neural Holography: A Flexible Framework for Holographic Near-eye Displays with Fast Heavily-quantized Spatial Light Modulators. In *ACM SIGGRAPH 2022 Conference Proceedings*. New York, NY, USA. https://doi.org/10.1145/3528233.3530734

[7] Robert L. Cook. 1986. Stochastic Sampling in Computer Graphics. *ACM Trans. Graph.* 5, 1 (jan 1986).

[8] Zvi Drezner and Horst W Hamacher. 2004. *Facility location: applications and theory.*

[9] Basile Fraboni, Antoine Webanck, Nicolas Bonneel, and Jean-Claude Iehl. 2022. Volumetric Multi-View Rendering. *Computer Graphics Forum* 41, 2 (2022).

[10] Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, et al. 2018. Arnold: A brute-force production path tracer. *ACM Trans. Graph.* 37, 3 (2018).

[11] Manu Gond, Emin Zerman, Sebastian Knorr, and Mårten Sjöström. 2023. LF-SphereNet: Real Time Spherical Light Field Reconstruction from a Single Omnidirectional Image. In *Proceedings of the 20th ACM SIGGRAPH European Conference on Visual Media Production* (London, UK) *(CVMP '23)*. New York, NY, USA, Article 10. https://doi.org/10.1145/3626495.3626500

[12] Ugur Gudelek, Erdem Sahin, and Atanas Gotchev. 2023. Perceptually Optimized Model for Near-Eye Light Field Reconstruction. In *2023 IEEE 25th International Workshop on Multimedia Signal Processing (MMSP)*. https://doi.org/10.1109/MMSP59012.2023.10337716

[13] Anne Juhler Hansen, Jákup Klein, and Martin Kraus. 2017. Light Field Rendering for Head Mounted Displays using Pixel Reprojection. In *VISIGRAPP (1: GRAPP)*.

[14] Hong Hua. 2017. Enabling focus cues in head-mounted displays. In *Imaging and Applied Optics 2017 (3D, AIO, COSI, IS, MATH, pcAOP)*. Optica Publishing Group.

[15] Julius Ikkala, Markku Mäkitalo, Tuomas Lauttia, Erwan Leria, and Pekka Jääskeläinen. 2022. Tauray: A Scalable Real-Time Open-Source Path Tracer for Stereo and Light Field Displays. In *SIGGRAPH Asia 2022 Technical Communications* (Daegu, Republic of Korea) *(SA '22)*. New York, NY, USA, Article 11. https://doi.org/10.1145/3550340.3564225

[16] James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986). https://doi.org/10.1145/15886.15902

[17] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.* 35, 6, Article 193 (dec 2016). https://doi.org/10.1145/2980179.2980251

[18] Leonard Kaufman and Peter J. Rousseeuw. 1987. Partitioning Around Medoids (Program PAM). *Finding Groups in Data: An Introduction to Cluster Analysis* (1987).

[19] Dongyeon Kim, Seung-Woo Nam, Suyeon Choi, Jong-Mo Seo, Gordon Wetzstein, and Yoonchan Jeong. 2024. Holographic Parallax Improves 3D Perceptual Realism. *ACM Trans. Graph.* 43, 4, Article 68 (jul 2024), 13 pages. https://doi.org/10.1145/3658168

[20] Gregory Kramida. 2016. Resolving the Vergence-Accommodation Conflict in Head-Mounted Displays. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (2016).

[21] Wallace Lages, Carlúcio Cordeiro, and Dorgival Guedes. 2008. A Parallel Multi-view Rendering Architecture. In *2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*. https://doi.org/10.1109/SIBGRAPI.2008.41

[22] Wallace Lages, Carlúcio Cordeiro, and Dorgival Guedes. 2009. Performance analysis of a parallel multi-view rendering architecture using light fields. *The Visual Computer* 25, 10 (2009).

[23] Leonardo Lanante Jr. 2023. Wi-Fi 8: Ultra High Reliability in the Unlicensed Bands. *Ofinno* (2023). https://ofinno.com/whitepaper/wi-fi-8-ultra-high-reliability-in-the-unlicensed-bands/

[24] Douglas Lanman and David Luebke. 2013. Near-Eye Light Field Displays. *ACM Trans. Graph.* 32, 6, Article 220 (nov 2013).

[25] Grigory Lazarev, Andreas Hermerschmidt, Sven Krüger, and Stefan Osten. 2012. LCOS spatial light modulators: trends and applications. *Optical Imaging and Metrology: Advanced Technologies* (2012).

[26] Erwan Leria, Markku Mäkitalo, and Pekka Jääskeläinen. 2024. Real-time Stereoscopic Image-parallel Path Tracing. *Electronic Imaging* 36, 13 (2024). https://doi.org/10.2352/EI.2024.36.13.ERVR-181

[27] Xue Liu, Xinzhu Sang, Xiao Guo, Shujun Xing, Yuanhang Li, Jinhui Yuan, and Binbin Yan. 2021. Real-time Super High Resolution Light Field Rendering with Multi-GPU Scheduling. In *2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*.

[28] Amazon Lumberyard. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/amazon-lumberyard-bistro

[29] Markku Mäkitalo, Erwan Leria, Julius Ikkala, and Pekka Jääskeläinen. 2023. Real-Time Light Field Path Tracing. In *Advances in Computer Graphics: 39th Computer Graphics International Conference, CGI 2022, Virtual Event, September 12–16, 2022, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 211–226. https://doi.org/10.1007/978-3-031-23473-6_17

[30] Markku J Mäkitalo, Petrus EJ Kivi, and Pekka O Jääskeläinen. 2020. Systematic Evaluation of the Quality Benefits of Spatiotemporal Sample Reprojection in Real-Time Stereoscopic Path Tracing. *IEEE Access* 8 (2020).

[31] Natalia D Mankowska, Anna B Marcinkowska, Monika Waskow, Rita I Sharma, Jacek Kot, and Pawel J Winklewski. 2021. Critical flicker fusion frequency: a narrative review. *Medicina* 57, 10 (2021), 1096.

[32] Morgan McGuire. 2017. Computer Graphics Archive.

[33] Yuta Miyanishi, Erdem Sahin, and Atanas Gotchev. 2019. Providing focus cues by light field displays: A review. In *European Light Field Imaging Workshop*.

[34] Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. 1994. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications* 14, 4 (1994).

[35] Carl Mueller. 1995. The Sort-First Rendering Architecture for High-Performance Graphics. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics* (Monterey, California, USA) *(I3D '95)*. New York, NY, USA. https://doi.org/10.1145/199404.199417

[36] Seung-Woo Nam, Dongyeon Kim, Suyeon Choi, Juhyun Lee, Siwoo Lee, Manu Gopakumar, Brian Chao, Gordon Wetzstein, and Yoonchan Jeong. 2024. Holographic Parallax. In *ACM SIGGRAPH 2024 Emerging Technologies* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 11, 2 pages. https://doi.org/10.1145/3641517.3664386

[37] M. Okutomi and T. Kanade. 1993. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 4 (1993), 353–363. https://doi.org/10.1109/34.206955

[38] Hyungman Park, Donald Fussell, and Paul Navrátil. 2022. Data-Aware Predictive Scheduling for Distributed-Memory Ray Tracing. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022). https://doi.org/10.1109/TVCG.2021.3114838

[39] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.).

[40] Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*.

[41] Elena Vasiou, Konstantin Shkurko, Ian Mallett, Erik Brunvand, and Cem Yuksel. 2018. A detailed study of ray tracing performance: render time and energy cost. *The Visual Computer* 34 (2018). https://api.semanticscholar.org/CorpusID:253894140

[42] Jakub Žádník, Markku Mäkitalo, Jarno Vanne, and Pekka Jääskeläinen. 2022. Image and Video Coding Techniques for Ultra-low Latency. *ACM Comput. Surv.* 54, 11s, Article 231 (sep 2022), 35 pages. https://doi.org/10.1145/3512342

[43] Ingo Wald, Milan Jaroš, and Stefan Zellmann. 2023. Data Parallel Multi-GPU Path Tracing using Ray Queue Cycling. In *Computer Graphics Forum*, Vol. 42.

[44] Hayato Watanabe, Takuya Omura, Naoto Okaichi, Masanori Kano, Hisayuki Sasaki, and Jun Arai. 2022. Full-parallax three-dimensional display based on light field reproduction. *Optical Review* 29 (2022), 366 – 374. https://api.semanticscholar.org/CorpusID:250251859

[45] Alfred Weber. 1909. Theory of industrial location. *San José State University Department of Economics: San Jose, CA, USA* (1909).

[46] Alexander Weinrauch, Stephan Lorbek, Wolfgang Tatzgern, Pascal Stadlbauer, and Markus Steinberger. 2023. Clouds in the Cloud: Efficient Cloud-Based Rendering of Real-Time Volumetric Clouds. In *High-Performance Graphics - Symposium Papers*, Jacco Bikker and Christiaan Gribble (Eds.). https://doi.org/10.2312/hpg.20231138

[47] Endre Weiszfeld. 1937. Sur le point pour lequel la Somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series* 43 (1937).

[48] Niko Wißmann, Martin Mišiak, Arnulph Fuhrmann, and Marc Erich Latoschik. 2020. Accelerated stereo rendering with hybrid reprojection-based rasterization and adaptive ray-tracing. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*.

[49] Lei Xiao, Anton Kaplanyan, Alexander Fix, Matthew Chapman, and Douglas Lanman. 2018. DeepFocus: learned image synthesis for computational displays.

ACM Trans. Graph. 37, 6, Article 200 (dec 2018). https://doi.org/10.1145/3272127.3275032

[50] Feng Xie, Petro Mishchuk, and Warren Hunt. 2021. Real Time Cluster Path Tracing. In *SIGGRAPH Asia 2021 Technical Communications* (Tokyo, Japan) *(SA ’21 Technical Communications)*. New York, NY, USA, Article 18. https://doi.org/10.1145/3478512.3488605

[51] Erfan Momeni Yazdi, Markku Mäkitalo, Julius Ikkala, and Pekka Jääskeläinen. 2023. TauBench 1.1: A Dynamic Benchmark for Graphics Rendering. arXiv:2305.04804 [cs.GR]

[52] Stefan Zellmann, Nate Morrical, Ingo Wald, and Valerio Pascucci. 2020. Finding Efficient Spatial Distributions for Massively Instanced 3-D Models. In *Eurographics Symposium on Parallel Graphics and Visualization*, Steffen Frey, Jian Huang, and Filip Sadlo (Eds.). https://doi.org/10.2312/pgv.20201070