Real-time 3D Visualization of Radiance Fields on Light Field Displays

JONGHYUN KIM*, NVIDIA, USA
CHENG SUN*, NVIDIA, USA
MICHAEL STENGEL*, NVIDIA, USA
MATTHEW CHAN, NVIDIA, USA
ANDREW RUSSELL, NVIDIA, USA
JAEHYUN JUNG, NVIDIA, USA
WIL BRAITHWAITE, NVIDIA, USA
SHALINI DE MELLO, NVIDIA, USA
DAVID LUEBKE, NVIDIA, USA

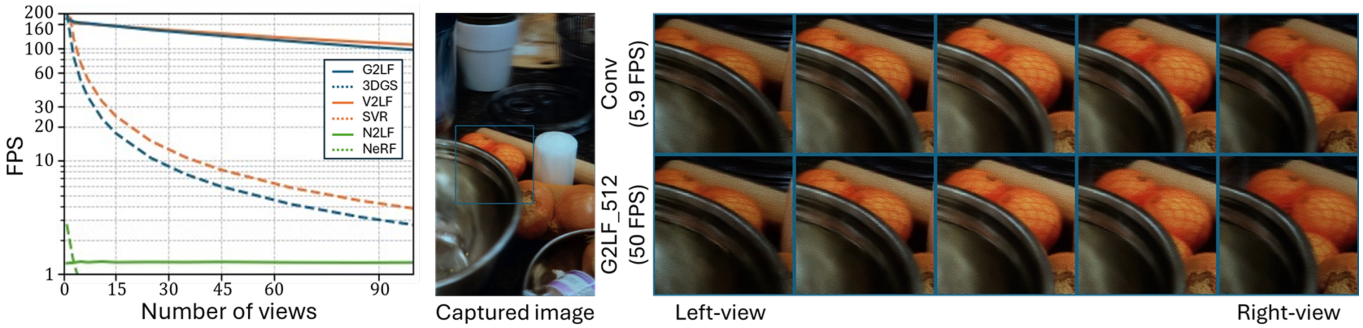

Fig. 1. We present a unified framework for real-time radiance field rendering on light field displays, supporting NeRFs (N2LF), 3D Gaussians (G2LF), and sparse voxels (V2LF) via a single-pass plane sweep. (Left) Render time comparison across view counts on the MipNeRF-360 dataset [Barron et al. 2022], measured at 512p on an RTX 5090. G2LF and V2LF maintain real-time performance (>60 FPS at N_{chunk}=64) beyond 90 views. (Right) Captured results from a Looking Glass Go display, using a motorized rotational stage, confirm high-quality 3D reconstruction up to 22× speedup over per-view rendering.

Radiance fields have revolutionized photo-realistic 3D scene visualization by enabling high-fidelity reconstruction of complex environments, making them an ideal match for light field displays. However, integrating these technologies presents significant computational challenges, as light field displays require multiple high-resolution renderings from slightly shifted viewpoints, while radiance fields rely on computationally intensive volume rendering. In this paper, we propose a unified and efficient framework for real-time radiance field rendering on light field displays. Our method supports a wide range of radiance field representations—including NeRFs, 3D Gaussian Splatting, and Sparse Voxels—within a shared architecture based on a single-pass plane sweeping strategy and caching of shared, non-directional components. The framework generalizes across different scene formats without retraining, and avoids redundant computation across views. We further demonstrate a real-time interactive application on a Looking Glass display, achieving 200+ FPS at 512p across 45 views, enabling seamless, immersive 3D interaction. On standard benchmarks, our method achieves up to 22× speedup compared to independently rendering each view, while preserving image quality.

1 INTRODUCTION

Recent advancements in radiance fields have significantly improved both the quantity and quality of 3D content. Radiance fields represent 3D scenes by encoding density and color values across spatial coordinates and view directions, enabling photorealistic reconstructions of complex scenes. Neural Radiance Fields (NeRF) have enabled continuous view synthesis from sparse input images, making it possible to reconstruct complex 3D scenes with high precision [Mildenhall et al. 2021; Müller et al. 2022]. Radiance fields now encompass a variety of representations for improving rendering efficiency for real-time applications, including rasterization-based methods [Kerbl et al. 2023; Sun et al. 2024] and explicit scene structures [Sun et al. 2022; Takikawa et al. 2021] that avoid dense sampling.

Like other 3D content, radiance fields are most effectively visualized using 3D displays. Their ability to represent complex 3D structures aligns naturally with the capabilities of light field displays, which optically reconstruct the light rays of 3D scenes. Recent commercially available light field displays offer high spatial and angular resolution, enabling immersive 3D visualization ([Leia Inc. 2025], [Sony Electronics Inc. 2025], [Looking Glass Factory Inc. 2025]). These displays provide binocular disparity and motion parallax, leveraging human depth perception to allow users to naturally perceive 3D structures. This integration, however, introduces significant computational challenges. First, light field displays require high-resolution rendering from many (45+) slightly shifted viewpoints. Second, generating views from radiance fields, even with fast rasterization-based methods like Gaussian Splatting, remains more computationally expensive than traditional graphics pipelines.

Light field displays fundamentally face substantial computational overhead in rendering due to their unique optical design. Unlike conventional single-view 2D displays, they require the generation

*contributed equally to this research

of multiple perspective views to reconstruct the full light field, necessitating a dense array of rays projected at precise angles. This significantly increases the computational burden compared to single-view 2D displays. Furthermore, precise optical alignment between the display panel and the lens array is critical; even minor angular or spatial misalignment during manufacturing can lead to incorrect ray-to-subpixel mappings, degrading visual quality (see section 3 for details). These misalignments demand a per-device calibration process to ensure accurate ray alignment, further complicating the rendering pipeline.

Radiance fields face inherent computational challenges, particularly due to their reliance on volumetric rendering, where density and color values must be evaluated along each ray. To reduce these costs, recent approaches employ explicit/hybrid data structures and rasterization-based pipelines to accelerate rendering, such as 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] and sparse voxels [Sun et al. 2024]. However, when targeting light field displays, these methods still require rendering many slightly shifted viewpoints, making repeated sampling inefficient and redundant. Addressing this redundancy is essential to enabling real-time radiance field rendering for light field applications.

In this paper, we present a unified framework for real-time radiance field rendering on light field displays. Our framework supports a wide range of radiance field representations—including NeRFs, 3DGS, and sparse voxels—within a unified rendering architecture. The core of our method is a single-pass plane sweeping strategy that enables efficient view synthesis while preserving image quality. We implement this framework in three variants: NeRF-to-light-field (N2LF), 3DGS-to-light-field (G2LF), and sparse-voxels-to-light-field (V2LF), corresponding to different input representations. We demonstrate the effectiveness of our approach through an interactive application running on a commercial light field display, enabling smooth and immersive real-time 3D visualization of radiance field. Our GL-based interactive demo achieves 228 FPS (22× faster compared to native quilt rendering, see section 5.3) for 45-view, 512×910 light field images on the BICYCLE scene using a single NVIDIA RTX 5090 graphics card. (see supplementary video)

The contributions of this paper are as follows:

- We propose a unified and efficient multi-view rendering framework for radiance fields for light field displays, supporting both implicit (NeRF) and explicit/hybrid (3DGS, sparse voxel) representations without retraining.
- Our method achieves real-time performance (>60 FPS for 90+ views, up to 228 FPS for 45 views) by minimizing redundant sampling through a single-pass plane sweeping strategy.
- We develop an OpenGL-based interactive 3DGS renderer for a commercial light field display, enabling real-time 3D visualization and achieving a 3.6× speedup over our Python/CUDA baseline.

2 RELATED WORK

Multi-view rendering. Techniques such as multi-view point splatting [Hübner et al. 2006], single-pass multi-view rendering [Hübner et al. 2007], and unstructured lumigraph rendering [Buehler et al. 2001], as well as hardware-level instancing methods [NVIDIA Corporation 2025; Unterguggenberger et al. 2020], reduce rendering cost across nearby views. However, these methods are tailored to traditional 3D content, such as meshes or point clouds, and are difficult to apply to radiance fields, where sharing computation across nearby views remains challenging. Our method addresses this by providing a unified solution that supports both implicit and explicit representations, enabling single-pass rendering of high-quality multi-view outputs.

Depth-guided view synthesis. Classical methods such as depth-image-based rendering [Fehn 2004], layered depth images [Shade et al. 1998], and multiplane images [Zhou et al. 2018] synthesize novel views by projecting RGB-D inputs onto proxy geometry or discrete depth layers. These efficient methods rely on given depth and fixed textures, often leading to disocclusion artifacts and limited geometric fidelity [Sun et al. 2010]. While our approach shares structural similarities, we operate directly on learned volumetric representations, enabling accurate reconstruction of complex view-dependent effects without requiring explicit depth inputs.

Radiance field representations. NeRF introduced continuous volumetric scene encoding for novel view synthesis [Mildenhall et al. 2021], prompting numerous efforts to accelerate rendering. Techniques include hash-based feature grids [Müller et al. 2022], space decomposition [Chen et al. 2022a; Reiser et al. 2021], and hierarchical training frameworks [Barron et al. 2022; Tancik et al. 2023]. More recently, explicit and hybrid representations such as 3DGS [Kerbl et al. 2023] and voxel-based rasterization [Sun et al. 2024, 2022; Takikawa et al. 2021] have enabled real-time rendering through rasterization-friendly pipelines. While these methods reduce the cost of rendering a single view, they do not directly address the inefficiency of rendering many closely related views, as required by multi-view displays.

Multi-view radiance field rendering. Rendering multiple views from radiance fields is computationally intensive due to high redundancy between adjacent viewpoints. Existing methods typically render each view independently, leading to inefficiencies in multi-view or light field applications [Rabia et al. 2024; Stengel et al. 2023; Tran et al. 2024]. Volume rendering with precomputed ray-to-subpixel mappings [Chen et al. 2022b; Ji et al. 2025; Yang et al. 2024] reduces overhead but requires per-device calibration for accurate rendering. In addition, these methods are not well suited to rasterization-based pipelines such as 3D Gaussian Splatting, where sorting and blending overheads hinder real-time performance despite known ray directions. Our method complements these efforts by minimizing redundant sampling across slightly shifted views using a unified framework that supports both implicit and explicit radiance field representations.

3 LIGHT FIELD 3D DISPLAY

Multi-view displays are designed to deliver binocular disparities at different viewpoints, providing motion parallax for viewer(s). The light rays from each microlens converge to specific viewpoints (Fig. 2, left). Such view separation is commonly achieved through optical elements, including parallax barriers [Lanman et al. 2010], lenticular

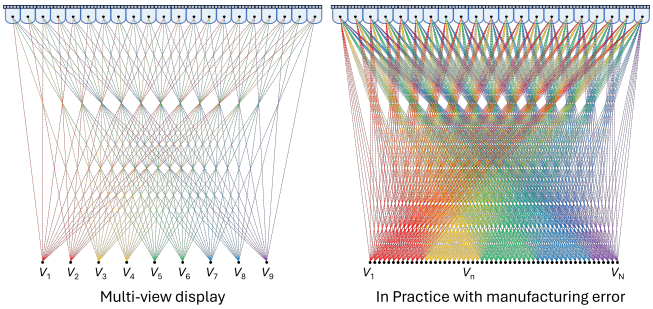

Fig. 2. Ray diagrams of light field displays from the top. *Left*: Multi-view display (9-view) and *right*: Dense-view display with a small manufacturing misalignment (0.00884°) in the lens array. A 1080p liquid crystal display paired with a non-slanted lens array is assumed for simplicity.

lenses [Ives 1931], or slanted lenticular lenses [Van Berkel 1999], attached to the display panel. The rendering process of multiple viewpoints involves placing perspective cameras in parallel at the designated viewing distance. After rendering all views, the base image (i.e., the encoded image shown on the panel) is interleaved from the rendered views.

Recent commercially available light field 3D displays often trade spatial resolution for higher angular resolution to achieve smoother motion parallax (i.e., increased the number of viewpoints). However, this design choice imposes a significant computational burden: while more views must be rendered, the display resolution remains fixed, resulting in substantial underutilization of the rendered data.

Additionally, manufacturing imperfections, such as angular and spatial misalignments between the lens array and display panel, further increase computational cost. Even a small angular misalignment, such as a half-subpixel offset (only 0.00884° for a 1080p LCD), can scramble all viewpoints (Fig. 2, right) [Kim et al. 2015]. Correcting these errors requires a per-device calibration process to estimate accurate view-to-subpixel mappings, ensuring geometrically correct 3D reconstruction. Although effective, this calibration further increases the number of rendered views and amplifies computational demands. As a result, modern light field displays often require 45 or more rendered views, posing significant challenges for real-time, interactive 3D rendering, particularly when dealing with radiance fields content.

Recent methods reduce the number of ray marches or rasterization steps by leveraging ray-to-subpixel mappings for efficient multi-view rendering on light field displays [Chen et al. 2022b; Ji et al. 2025; Yang et al. 2024]. While these approaches improve rendering efficiency, they still rely on per-device calibration and have yet to generalize to diverse radiance field representations with real-time performance.

To address these limitations, we propose a unified framework for radiance field rendering on light field displays. Our method reduces computational redundancy by minimizing repeated sampling across minimally shifted viewpoints, while maintaining overall perceived image quality, and is compatible with various radiance field representations. Detailed algorithmic descriptions follow in the next section.

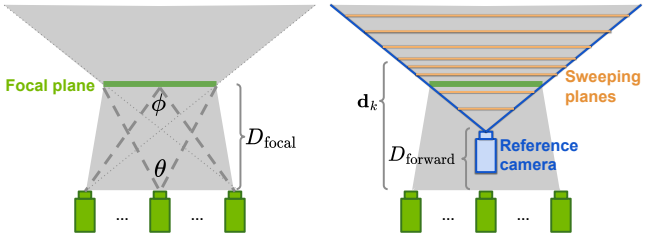

Fig. 3. Illustration of the 3D display parameterization and the forward sweeping planes. θ is base camera field of view and ϕ is 3D display viewing angle. The reference camera is positioned to cover the entire visible volume behind the focal plane.

4 RADIANCE FIELDS TO LIGHT FIELD

We first give an overview of our efficient radiance fields to light fields rendering pipeline in Sec. 4.1 and introduce the shared components in Sec. 4.2 and Sec 4.3. Later, we adapt our algorithm to various 3D representations commonly used in radiance fields reconstruction and novel-view synthesis. The representation-specific adaptations are described in Sec. 4.4, 4.5, and 4.6.

4.1 Rendering pipeline overview

Given a radiance field and a 3D display viewing setup, our goal is to render a set of $V = (V_x \times V_y)$ perspective viewpoints derived from the setup, called light field quilts as illustrated in Fig. 2. A conventional method applies a radiance field renderer V times separately. However, such a simple approach can slow down rendering up to V times, which prevents interactive experiences even for a 200 FPS renderer for a common V=45 light field display.

Our strategy is to reduce computation by approximating the volume visible to the 3D display by a series of forward-sweeping planes. Each plane represents a disjoint frustum of the original volume. We can then composite the sweeping planes into the V quilt views via an operation that we call swizzle blending, instead of traversing through the original 3D representation. A single pixel lookup on the planes corresponds to the rendering result of a ray segment from the original radiance field, where the latter is more expensive to compute. Thus the overall rendering time can be largely reduced as long as we can also render the sweeping planes efficiently. Next, we introduce the algorithm for sweeping planes in Sec. 4.2 and swizzle blending in Sec. 4.3.

4.2 Sweeping planes rendering

Let's first define the volume of interest of a 3D display setup, which is illustrated in Fig. 3's left panel. We can imagine the 3D display as a "window" (called focal plane) for the viewers to look into the virtual world. The center location of the focal plane is defined by a *b*ase camera and a camera-to-plane distance as D_{focal}. The size of the focal plane is derived from the base camera's field of view θ_x, θ_y. The viewing angles of the the 3D display are defined by ϕ_x, ϕ_y. Finally, the visible volume to the 3D display is determine by the maximum viewing angles.

To create a series of forward-sweeping planes, we use a perspective *reference* camera, which is adjusted to cover the maximum

viewing angle as shown in Fig. 3's right panel. To this end, we move forward from the base camera's pose by a distance:

$$D_{\text{forward}} = \max_{k\in\{x,y\}} D_{\text{focal}} \cdot \frac{\tan(0.5\phi_k)}{\tan(0.5\phi_k)+\tan(0.5\theta_k)}\,, \tag{1}$$

and set the field of view of the reference camera as:

$$\theta'_{k\in\{x,y\}} = 2\cdot\arctan\left(\frac{D_{\text{focal}}\cdot\tan(0.5\theta_k)}{D_{\text{focal}}-D_{\text{forward}}}\right). \tag{2}$$

By doing so, we ensure that the volume to display behind the focal plane is all visible by the reference camera. One limitation is that some area in-between the V viewing cameras and the focal plane are ignored. As a workaround, we simply introduce a hyperparameter D_{shift} to move the camera backward: $D_{\text{forward}} \leftarrow D_{\text{forward}} - D_{\text{shift}}$. Future work can improve this by adding another mirrored reference camera behind the focal plane and rendering in the "backward" direction, while we find our existing simple workaround can already achieve satisfactory results.

Finally, the sweeping planes are generated by rendering the scene primitive's chunks or volume chunks into the perspective reference camera. We detail the chunking and rendering methods of different 3D representations in the later sections. The rendered forward-sweeping planes are denoted as $\mathbf{C}\in\mathbb{R}^{N_{\text{chunk}}\times(P_s\cdot N_y)\times(P_s\cdot N_x)\times 3}$ for RGB colors and $\mathbf{T}\in\mathbb{R}^{N_{\text{chunk}}\times(P_s\cdot N_y)\times(P_s\cdot N_x)}$ for transmittances, where N_{chunk} is the total number of chunks, $N_x\times N_y$ is a single-view quilt resolution, and P_s is a resolution scaling hyperparameter of the planes. The color planes imply that the view-directional color is only accurate for the central view of the 3D display. We also experiment with spherical harmonics instead, while we find the improvement is marginal with much more compute and memory usage.

4.3 Swizzle blending

We can now render the final quilts by alpha composition, efficiently using the rendered planes. The quilts $\mathbf{Q}\in\mathbb{R}^{V_y\times V_x\times N_y\times N_x\times 3}$ is a 2D array of perspective views formed by moving the base camera along its horizon and vertical directions. The camera offsets (Δ_x,Δ_y) are linearly interpolated in the angular domain of viewing angles, and their principal points (c_x,c_y) always aim toward the focal plane's center such that the $N_y\times N_x$ rays from all quilt views converge at the focal plane. The equations for the x component are given as:

$$\rho_j = \phi_x\cdot\left(\frac{j-1}{V_x-1}-\frac{1}{2}\right),\ \Delta_x = D_{\text{focal}}\cdot\tan(\rho_j),\ c_x = \frac{\tan(\rho_j)}{\tan(0.5\theta_x)}, \tag{3}$$

where $j\in[1,V_x]$ is the column index to the quilt views, and c_x is in normalized image domain (i.e., image border at ±1). Given a normalized pixel x-coordinates $u\in[0,1]$ on the j-th column of quilts, their projected coordinate to the k-th forward-sweeping planes at distance $\mathbf{d}_k$ is:

$$u' = \frac{\overbrace{(D_{\text{focal}}-\mathbf{d}_k)\cdot\tan(\rho_j)}^{\text{Coordinate of the principal point.}} + \overbrace{\mathbf{d}_k\cdot\tan(0.5\theta_x)\cdot u}^{\text{Offset from the principal point.}}}{(\mathbf{d}_k-D_{\text{forward}})\cdot\tan(0.5\theta'_x)}. \tag{4}$$

The above equations can be extended to y component, similarly.

We use Eq. 4 to project a quilt pixel onto the sweeping planes and sample $\mathbf{C}$ and $\mathbf{T}$ via either bilinear or nearest-neighbor interpolation. The sampled series of colors c_k and transmittance values T_k are blended into a final pixel color with:

$$C = \sum_{k=1}^{N_{\text{chunk}}}\left(\prod_{j=1}^{j<k} T_j\right)\cdot c_k\,. \tag{5}$$

Note that the color c_k here is already weighted by alpha opacity. We also find using 8 bits unsigned integer to store $\mathbf{C}$ and $\mathbf{T}$ leading to similar blending quality comparing to using 32 bits float. Thus we use 8 bits by default, which leads to much less memory usage and faster rendering.

In the following, we introduce the representation-specific adaptions of our algorithm.

4.4 G2LF – 3D Gaussians to Light Field

One advantage of a primitive-based representation is that we can have a similar number of primitives inside each chunk by quantile binning. For applying quantile binning with 3DGS, we filter Gaussians inside the view frustum of the reference camera and set the chunking distances at N_{chunk}+1 linearly spaced percentiles using the distances of Gaussian centers to the reference camera. The plane distance $\mathbf{d}_k$ of k-th plane is set to the median Gaussian distance of that chunk. We extend the efficient CUDA rasterizer by [Kerbl et al. 2023] to perform 3D tiling instead of the original 2D tiling with an additional dimension for the chunks. A Gaussian is assigned to a tile by its patch index and chunk index. The Gaussians in each 3D tile are sorted and rendered in parallel, which finally produces the color and transmittance forward-sweeping planes: $\mathbf{C}$ and $\mathbf{T}$. A pipeline visualization and pseudocode of the entire rendering procedure is given by Fig. 4 and Algo. 1.

The original 3DGS uses a hard-coded antialiasing filter in pixel-size unit. Specifically, the variance of the projected 2D Gaussian on the screen space is always dilated by 0.3 pixel. This causes a mismatch between the conventional rendering and the sweeping-planes-based rendering from our reference camera with different plane resolution scaling factors P_s. We use an adaptive filtering strength to align our rendering with the conventional rendering by:

$$s' = s\cdot\left(\frac{D_{\text{focal}}\cdot\tan(0.5\theta_x)}{(D_{\text{focal}}-D_{\text{forward}})\cdot\tan(0.5\theta'_x)}\cdot P_s\right)^2, \tag{6}$$

where $s=0.3$ is the hard-coded number in the original 3DGS implementation. The equation yields larger filter strength (in the screen space of the reference camera) when the pixel size of the reference camera on the focal plane is smaller than the conventional one.

4.5 V2LF – Sparse Voxels to Light Field

The rendering of sparse voxels follow the same principle as the rendering of 3DGS thanks to its primitive property. We also extend the CUDA-based sparse voxel rasterizer (SVR) [Sun et al. 2024] in a similar way as used for adapting to 3DGS. Instead of pre-filtering the primitives, SVR employs supersampling with anti-aliased downsampling to tackle aliasing issue. However, resizing the sweeping planes can be slow and double the GPU memory usage, especially with large N_{chunk} or high P_s. As an alternative, we disable the supersampling and apply a low-pass Gaussian filter on the sweeping planes instead, which is implemented in CUDA and performs filtering inplace without allocating extra memory.

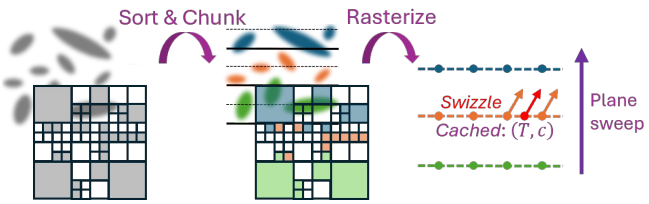

Fig. 4. Illustration of the G2LF and V2LF algorithm. The Gaussians and Voxels primitive are first sorted along cameras z-axis and grouped into chunks such that each chunk has similar number of primitives. Primitives in each chunk are rasterized onto their middle plane, producing a series of transmittance and color planes. We use swizzle blending to accumulate the plane values for each quilt pixel, enabling efficient rendering of the light field quilt.

Algorithm 1 G2LF/V2LF

P - Input Gaussians or Voxels primitives
H - Quilts and camera parameters
Q, T - Output light field quilt rgb and transmittance buffer

$\mathbf{d}' \leftarrow$ CulledDepth(P) ▷ GS/Voxels z distances to cam.
$\mathbf{d} \leftarrow$ FindQuantile($\mathbf{d}'$) ▷ N_{chunk} plane distances
$\{P'_k\} \leftarrow$ Sort_and_Chunk($P, \mathbf{d}$) ▷ Sort & Chunk GS/Voxels
Parallel for all k **do** ▷ Rasterize chunks to midplane
    $\mathbf{T}_k, \mathbf{C}_k \leftarrow$ Rasterize(H, P'_k) ▷ Cache grid values
$Q, T \leftarrow$ Initialize_buffers()
for k in $1...N_{\text{chunk}}$ **do** ▷ Swizzle blending
    $U \leftarrow$ Quilt2PlaneCoordinate($H, \mathbf{d}_k$) ▷ Eq.4
    $c \leftarrow$ interpolate($U, \mathbf{C}_k$)
    $t \leftarrow$ interpolate($U, \mathbf{T}_k$)
    $Q, T \leftarrow (Q + T \odot c), (T \odot t)$ ▷ Eq.5
end for

4.6 N2LF – NeRF to Light Field

We also adapt our rendering algorithm to fully volumetric representations like NeRF. We use the same quantile binning strategy as employed for 3DGS and sparse voxels to avoid the manual effort to tune the chunking positions for different scenes and different viewpoints. To this end, we use the coarse network in NeRF to render a roughly estimated depth map from the base camera view first and quantile the depth points to determine the chunking positions. To render sweeping planes from reference cameras, we ablate the occlusion term when doing the hierarchical importance sampling along a ray so that the occluded region can still be sampled. Points colors and alphas from the final round of the sampling are accumulated into different chunks based on their sampling positions. Despite the effort, we still observe apparent degradation in quality of N2LF comparing to using the conventional quilts rendering with NeRF. We hypothesize that the main reason is due to the fact that the importance sampler is never trained to render sweeping planes for quilts. As a result, the sampler parameterized by neural networks encounters severe out of distribution issues in the case of N2LF. Designing a specific training method to address this is out of our current scope and we leave it for future work.

Table 1. Light field quilt rendering results comparison on the Mip-NeRF360 dataset using various algorithms and their ablations. The results are averaged on 9 scenes where we sample 4 viewpoints to render light field quilts for each scene (quilt resolution set to V_x=45, V_y=1, N_x=N_y=512). Nerfacto [Mildenhall et al. 2021], 3DGS [Kerbl et al. 2023], and SVR [Sun et al. 2024] are used for the baselines whose rendered quilts are served as the ground truth for N2LF, G2LF, and V2LF, respectively. The FPS indicate the rendering speed of the entire quilts measured on a NVIDIA RTX 5090.

Method	FPS↑	LPIPS↓	PSNR↑	SSIM↑
Nerfacto	0.1	serve as ground truth		
N2LF (N_{chunk}=64)	1.2	0.360	20.23	0.536
N2LF (N_{chunk}=256)	1.2	0.324	21.01	0.577
N2LF (N_{chunk}=64, P_s=1.5)	0.6	0.280	20.31	0.545
N2LF (N_{chunk}=256, P_s=1.5)	0.6	0.244	21.13	0.589
3DGS	5.9	serve as ground truth		
G2LF (N_{chunk}=64)	127	0.311	24.89	0.749
G2LF (N_{chunk}=64, P_s=2)	109	0.129	26.96	0.820
G2LF (N_{chunk}=512)	50	0.277	26.89	0.831
G2LF (N_{chunk}=512, **Bi.**)	24	0.156	28.78	0.875
G2LF (N_{chunk}=512, P_s=2)	34	0.084	30.71	0.924
G2LF (N_{chunk}=512, P_s=2, **Bi.**)	19	0.065	33.31	0.950
G2LF (N_{chunk}=512, P_s=3, **Bi.**)	15	0.053	34.77	0.958
SVR	8.2	serve as ground truth		
V2LF (N_{chunk}=64)	133	0.344	23.63	0.673
V2LF (N_{chunk}=64, P_s=2)	104	0.153	26.97	0.793
V2LF (N_{chunk}=512)	48	0.302	25.57	0.767
V2LF (N_{chunk}=512, **Bi.**)	26	0.182	28.51	0.845
V2LF (N_{chunk}=512, P_s=2)	26	0.099	31.06	0.908
V2LF (N_{chunk}=512, P_s=2, **Bi.**)	17	0.072	33.87	0.938
V2LF (N_{chunk}=512, P_s=3, **Bi.**)	11	0.055	35.54	0.950

5 RESULTS

5.1 Evaluation

Dataset. We use the four indoor and five outdoor unbounded real-world scenes from the Mip-NeRF360 dataset for evaluation. We use all the train-set cameras for training and sample four view points from the test set for evaluating the rendering quality.

Base models reproduction. We used Nerfacto [Tancik et al. 2023], 3DGS [Kerbl et al. 2023], and SVR [Sun et al. 2024] as the baseline models. We directly train with their default hyperparameters on the Mip-NeRF360 dataset. The standard perspective novel-view results are (LPIPS 0.379, PSNR 23.64, SSIM 0.678) for Nerfacto, and (LPIPS 0.215, PSNR 27.53, SSIM 0.816) for 3DGS, and (LPIPS 0.186, PSNR 27.36, SSIM 0.822) for SVR. The proposed N2LF, G2LF, and V2LF are evaluated based on the trained models.

Results on light field quilt. In Fig. 1 (left), the render FPS for varying number of views is presented using different algorithms. Our G2LF and V2LF algorithms achieve real-time performance (>30 FPS) even for over 90 views, enabling an interactive 3D visualization of the scene on light field displays.

Table 1 presents a quantitative comparison of light field quilts (V_x=45, V_y=1, N_x=N_y=512) rendered using various algorithms and their ablations. As we do not have the ground truth images for the

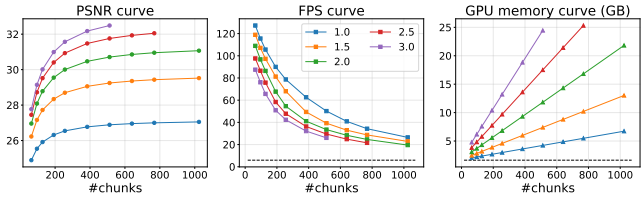

Fig. 5. Performance curves of G2LF under various N_{chunk} (x-axis) and P_s (colors). Some curves have fewer data points due to OOM. The black dash lines are the FPS and memory usage of the conventional 3DGS rendering. More numbers of planes and higher plane resolution both consistenly improve quality with the cost of more compute and memory usage. The quality improvement is about saturated at around N_{chunk}=400. Higher P_s keep improving the quality but memory usage also increases faster.

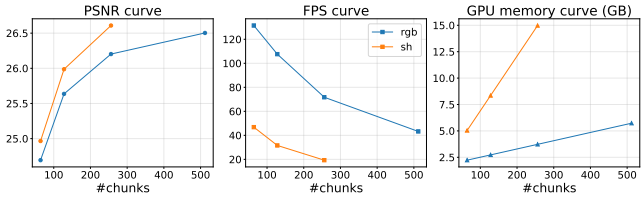

Fig. 6. Comparison of swizzle blending with color or spherical harmonic (SH) for more accurate view-dependent colors. The quality improvement is relatively minor comparing to using more chunks or higher plane resolution (Fig. 5). However, the FPS drop down and the increase in memory usage is significant. We thus disable this feature as the default setup.

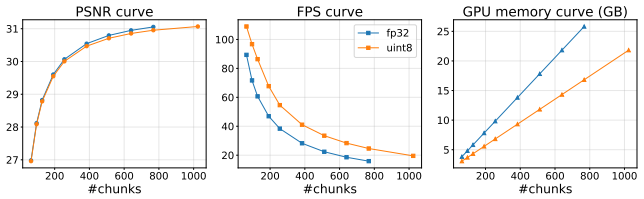

Fig. 7. Comparison of using uint8 and fp32 to store the sweeping planes of G2LF (P_s=2). The PSNR is almost identical while the run time and memory usage is largely reduced. The uint8 have one more data point with more N_{chunk} thanks to its less memory usage while fp32 encouters OOM.

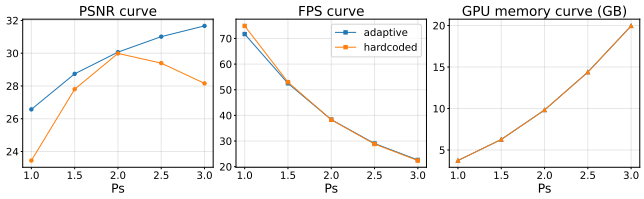

Fig. 8. Ablation of using adaptive filter strength or the original hard-coded one in G2LF (N_{chunk}=256). The hard-coded one encounters a severe anti-aliasing filter mismatch when changing the sweeping plane resolution. As a result, the quality even degraded with higher P_s. The adaptive filter strength solves this issue so the quality can keep improving with higher P_s. FPS and memory usage is almost the same for the two version.

quilts, we use the rendering results from the conventional method to serve as the ground truth for N2LF, G2LF, and V2LF. As discussed, our method encounters severe out-of-distribution issues with NeRF's importance sampler, which is parameterized by an MLP. This results in noticeable degradation of N2LF, despite being several times faster. On the other hand, we demonstrate faithful light field rendering results with G2LF and V2LF, comparable to their slower conventional counterparts: 3DGS and SVR. The number of chunks N_{chunk}, plane resolution scale P_s, and interpolation methods (nearest-neighbor as default; 'Bi.' for bilinear) are the most important knobs, which trades speed for quality. With moderate rendering quality drops of G2LF and V2LF, we achieve interactive frame rates for light field quilts rendering.

Increasing the resolution in z (N_{chunk}) and xy (P_s) camera axis can approach baseline quality at increasing computational cost. As an example, the rendered results in Fig. 12 confirm that our algorithms produce accurate perspective views in real time (see supplementary materials for more rendered results). Increasing the number of chunks trades FPS for improved render quality. Additionally, sampling at a 1.5x higher grid resolution per plane (indicated as 'highest') improves image quality by reducing interpolation errors during the *Swizzle*. Performance curves in Fig. 5 show extensive variations of G2LF with different number of chunks and plane resolution scales, both of which consistently mitigate difference with the slow conventional rendering. We show comprehensive ablation studies of various important aspects using G2LF and discuss their results in the captions—ablating spherical harmonic in Fig. 6, comparing uint8 and fp32 in Fig. 7, showing affect of adaptive anti-aliasing filter in Fig. 8, comparing different interpolations in Fig. 9, and different quilts setup in Fig. 10.

5.2 Captured Results

We captured the displayed light field results for several algorithm variants using a Looking Glass Go display and a Google Pixel 9 Pro (f/2.8, 1/50s, ISO 100), as shown in Fig. 14. The display was mounted on a motorized rotational stage, and viewpoints were uniformly sampled across the horizontal range (−18.5° to +17.5°) at 1.5° intervals. The brightness of the captured images was adjusted linearly for visualization. The video results confirm that our method enables high-quality 3D reconstruction with accurate angular consistency across views (see supplementary video). In practice, artifacts visible in individual rendered views are less noticeable during actual use, as multiple views blend perceptually within the viewer's pupil. See supplementary materials for more captured results.

5.3 Real-Time Interactive 3D Visualization Demo

We demonstrate the application of our method in a dynamic, interactive 3D visualization on a light field display. Fig. 11 showcases the demo running on a desktop equipped with a single NVIDIA RTX 5090 and a Looking Glass 16" Light Field Display. In comparison to a traditional 2D display, our real-time G2LF implementation allows users to perceive depth directly through binocular cues, providing a more intuitive and immersive interaction with 3DGS content. Our interactive application is implemented in OpenGL/C++ to make use of texture filtering and blending in hardware during the swizzle operation. We use 8-bit RGBA framebuffers for both slice buffers

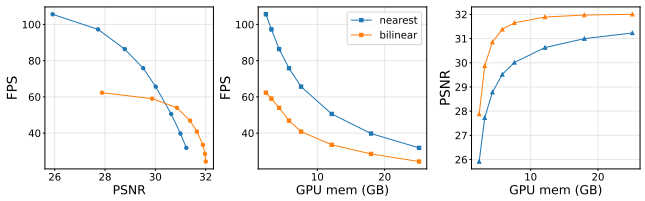

Fig. 9. Comparing bilinear and nearest-neighbor interpolation on swizzle blending with N_{chunk}=64 and $P_s \in [1, 6]$. As expected, bilinear interpolation achieves better quality but with slower FPS under the same amount of GPU memory usage comparing to nearest-neighbor interpolation. There is a crossover point in left figure at around 30.5 PSNR. To render in lower quality, nearest-neighbor is faster while bilinear becomes faster for achieving higher PSNR. This is because nearest-neighbor interpolation need higher P_s to achieve the same quality as bilinear, which becomes the dominant factor of computation with high P_s. Theoretically, both versions should achieve the same PSNR with zero FPS with infinite P_s.

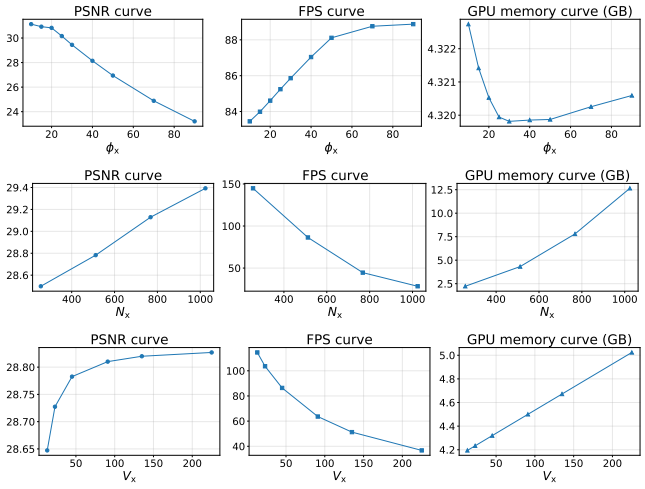

Fig. 10. Result of G2LF for different quilt setups. The default setup is P_s=2, N_{chunk}=128, ϕ_x=35, ϕ_y=0, N_x=N_y=512, V_x=45, V_y=1 with nearest interpolation and uint8. (Top) For different viewing angles ϕ_x, the FPS and memory usage difference is trivial, while the quality keep going down for covering more extreme angles. (Middle) The FPS and memory is about proportional to the resolution of each view on quilt. The quality difference is relatively minor with slightly better results when rendering higher resolution quilt views. (Bottom) Similarly, increasing the number of quilt views increases runtime and memory, while PSNR remains similar due to denser sampling within the same viewing range.

and the 45-views 4K x 4K quilt. Our OpenGL implementation mostly follows our description in Alg. 1. One difference is that we switch the order of quantile computation and sorting to sort all primitives in one pass upfront. From the rendered quilt we apply interlacing provided by the LookingGlass Bridge SDK before sending the frame to the swap chain.

The OpenGL implementation achieves comparable image quality to the Python/CUDA implementation presented in Tab. 1, but

Fig. 11. Real-time interactive 3D radiance field demonstration on a Looking Glass light field display, enabling users to dynamically change viewpoints and visualize radiance fields in a 3D space (see supplementary video).

achieves better computational performance due to higher GPU utilization. Compared to the Python/CUDA baseline, the speedup is approximately 3.6× on average across tested scenes (N_{chunk}=64, P_s=1, measured on NVIDIA RTX 5090). Our GL implementation scales well across varying number of chunks. In our tests N_{chunk}=64 to N_{chunk}=128 provide a sweet spot in terms of performance. Although higher numbers of chunks result in slightly higher image quality metrics the quality increase diminishes when perceived on the light field display (see supplementary video). Lower values than 32 slices can result in noticeable discrete depth layers for surfaces close to the camera and should be avoided.

6 DISCUSSION

Framework Versatility. Our unified framework supports a wide range of radiance field representations, including all existing and future NeRF variants, as well as explicit formats like 3D Gaussians and voxels, without the need for retraining. It is compatible with all types of light field displays, from 3D displays with head-tracked rendering to integral imaging systems offering vertical parallax. Unlike methods limited to a fixed set of view directions, our slice-based approach enables fast reconstruction from arbitrary viewpoints within the supported viewing angle.

Limitations and Future Work. One of the main limitations of our approach is the increased memory usage due to the intermediate slice-based representation, which is more demanding than direct rendering methods. Future work may explore reducing this memory footprint, for example by employing hardware-accelerated compression or more compact data layouts. An alternative direction involves investigating implicit representations of view-dependent slice information, which could further improve storage efficiency while maintaining rendering quality. Additionally, blending performance is inherently tied to the number and resolution of slice planes. While more slices enable finer volumetric detail, they also incur higher computational cost. Since the ultimate output is constrained by the characteristics of physical light field displays (e.g., limited angular or depth resolution), future optimization could benefit from adapting the slice structure to better match the display-specific capabilities and perceptual requirements.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5470–5479.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. 2001. Unstructured Lumigraph Rendering. In *SIGGRAPH*. ACM, 425–432. https://doi.org/10.1145/383259.383309

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022a. TensoRF: Tensorial Radiance Fields. In *ECCV*. 333–350. https://doi.org/10.1007/978-3-031-19824-3_21

Shuo Chen, Binbin Yan, Xinzhu Sang, Duo Chen, Peng Wang, Zeyuan Yang, Xiao Guo, and Chongli Zhong. 2022b. Fast virtual view synthesis for an 8k 3d light-field display based on cutoff-nerf and 3d voxel rendering. *Optics Express* 30, 24 (2022), 44201–44217.

Christoph Fehn. 2004. Depth-Image-Based Rendering (DIBR), Compression, and Transmission for a New Approach on 3D-TV. In *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, Vol. 5291. 93–104. https://doi.org/10.1117/12.524762

Thomas Hübner, Yanci Zhang, and Renato Pajarola. 2006. Multi-View Point Splatting. In *GRAPHITE*. ACM, 285–294. https://doi.org/10.1145/1174429.1174479

Thomas Hübner, Yanci Zhang, and Renato Pajarola. 2007. Single-Pass Multi-View Rendering. *IADIS Int. J. on Computer Science and Information Systems* 2, 2 (2007), 122–140.

Herbert E Ives. 1931. Optical properties of a Lippmann lenticulated sheet. *JOSA* 21, 3 (1931), 171–176.

Luyu Ji, Xinzhu Sang, Shujun Xing, Xunbo Yu, Binbin Yan, and Jiahui Yang. 2025. Text-driven light-field content editing for three-dimensional light-field display based on Gaussian splatting. *Optics Express* 33, 1 (2025), 954–971.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Jonghyun Kim, Chang-Kun Lee, Jong-Young Hong, Changwon Jang, Youngmo Jeong, Jiwoon Yeom, and Byoungho Lee. 2015. Calibration method for the panel-type multi-view display. *Journal of the Optical Society of Korea* 19, 5 (2015), 477–486.

Douglas Lanman, Matthew Hirsch, Yunhee Kim, and Ramesh Raskar. 2010. Content-adaptive parallax barriers: optimizing dual-layer 3D displays using low-rank light field factorization. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–10.

Leia Inc. 2025. Leia Official Website. https://www.leiainc.com/ Accessed: January 2025.

Looking Glass Factory Inc. 2025. Looking Glass Factory Official Website. https://lookingglassfactory.com/ Accessed: January 2025.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

NVIDIA Corporation. 2025. VRWorks - Multi-View Rendering (MVR). https://developer.nvidia.com/vrworks/graphics/multiview Accessed: May 2025.

Sédick Rabia, Guillaume Allain, Rosalie Tremblay, and Simon Thibault. 2024. Orthoscopic elemental image synthesis for 3D light field display using lens design software and real-world captured neural radiance field. *Optics Express* 32, 5 (2024), 7800–7815.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding Up Neural Radiance Fields with Thousands of Tiny MLPs. In *ICCV*. 14335–14345. https://doi.org/10.1109/ICCV48922.2021.01408

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 231–242.

Sony Electronics Inc. 2025. Spatial Reality Display by Sony. https://pro.sony/ue_US/products/spatial-reality-displays/3d-professional-images Accessed: January 2025.

Michael Stengel, Koki Nagano, Chao Liu, Matthew Chan, Alex Trevithick, Shalini De Mello, Jonghyun Kim, and David Luebke. 2023. AI-mediated 3D video conferencing. In *ACM SIGGRAPH 2023 Emerging Technologies*. 1–2.

Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. 2024. Sparse Voxels Rasterization: Real-time High-fidelity Radiance Field Rendering. *arXiv preprint arXiv:2412.04459* (2024).

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5459–5469.

Wenxiu Sun, Lingfeng Xu, Oscar C Au, Sung Him Chui, and Chun Wing Kwok. 2010. An overview of free view-point depth-image-based rendering (DIBR). In *APSIPA Annual Summit and Conference*. 1023–1030.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11358–11367.

Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. 2023. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–12.

Phong Tran, Egor Zakharov, Long-Nhat Ho, Anh Tuan Tran, Liwen Hu, and Hao Li. 2024. VOODOO 3D: Volumetric Portrait Disentanglement for One-Shot 3D Head Reenactment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10336–10348.

Johannes Unterguggenberger, Bernhard Kerbl, Markus Steinberger, Dieter Schmalstieg, and Michael Wimmer. 2020. Fast Multi-View Rendering for Real-Time Applications.. In *EGPGV@ Eurographics/EuroVis*. 13–23.

Cees Van Berkel. 1999. Image preparation for 3D LCD. In *Stereoscopic Displays and Virtual Reality Systems VI*, Vol. 3639. SPIE, 84–91.

Zongyuan Yang, Baolin Liu, Yingde Song, Lan Yi, Yongping Xiong, Zhaohe Zhang, and Xunbo Yu. 2024. DirectL: Efficient Radiance Fields Rendering for 3D Light Field Displays. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–19.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

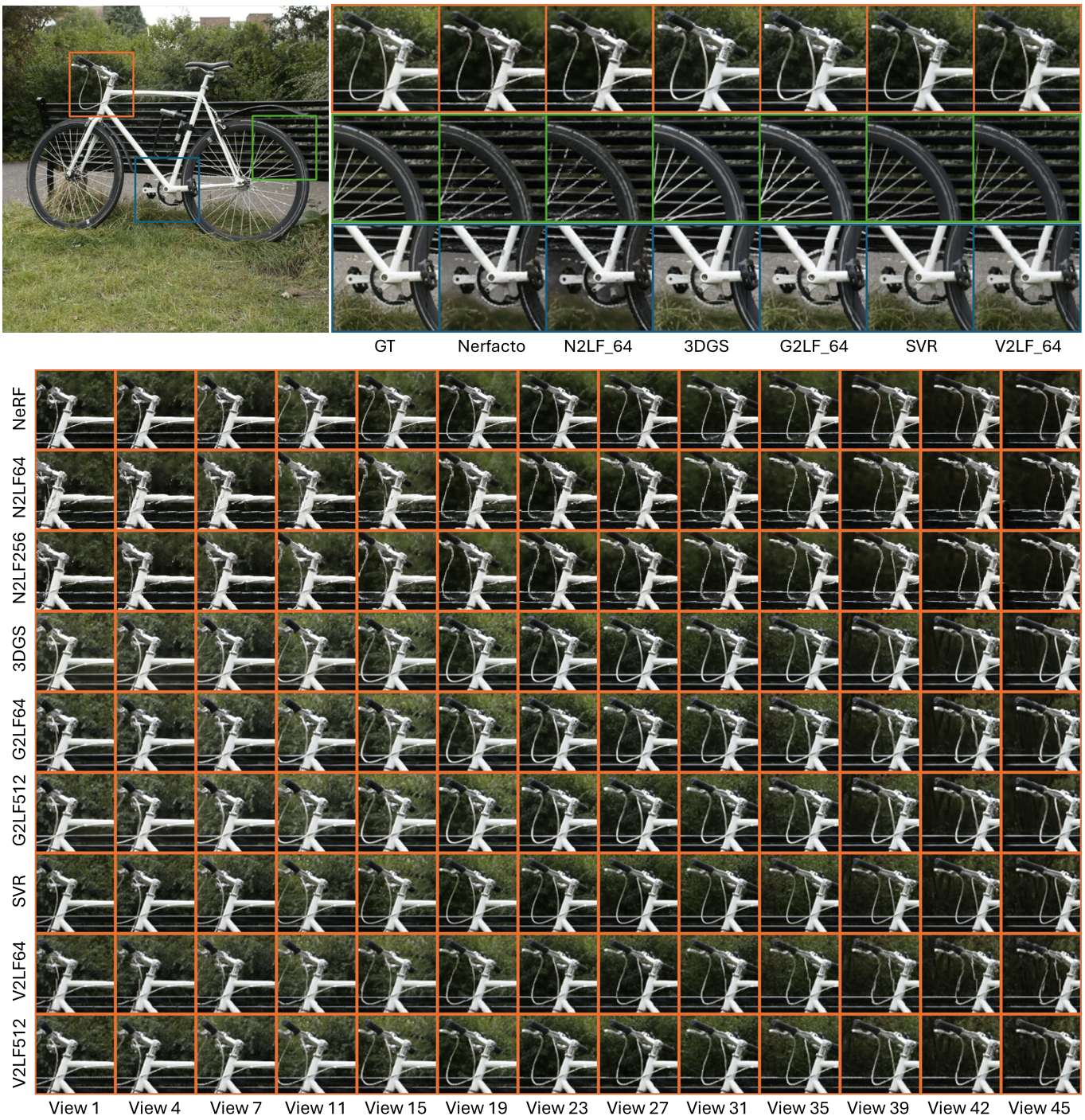

Fig. 12. Evaluation of the rendered Bicycle scene from the MipNeRF-360 dataset [Barron et al. 2022]. (Top) Visual comparisons between the ground truth, baseline models, and our proposed methods. (Bottom) Light field quilt rendering results using different algorithms.

Fig. 13. Close-up comparison of the BONSAI and GARDEN scene at the (top) left-most (View 45) and (bottom) right-most (View 1) viewing angle. Increasing the number of chunks minimizes depth discontinuity errors.

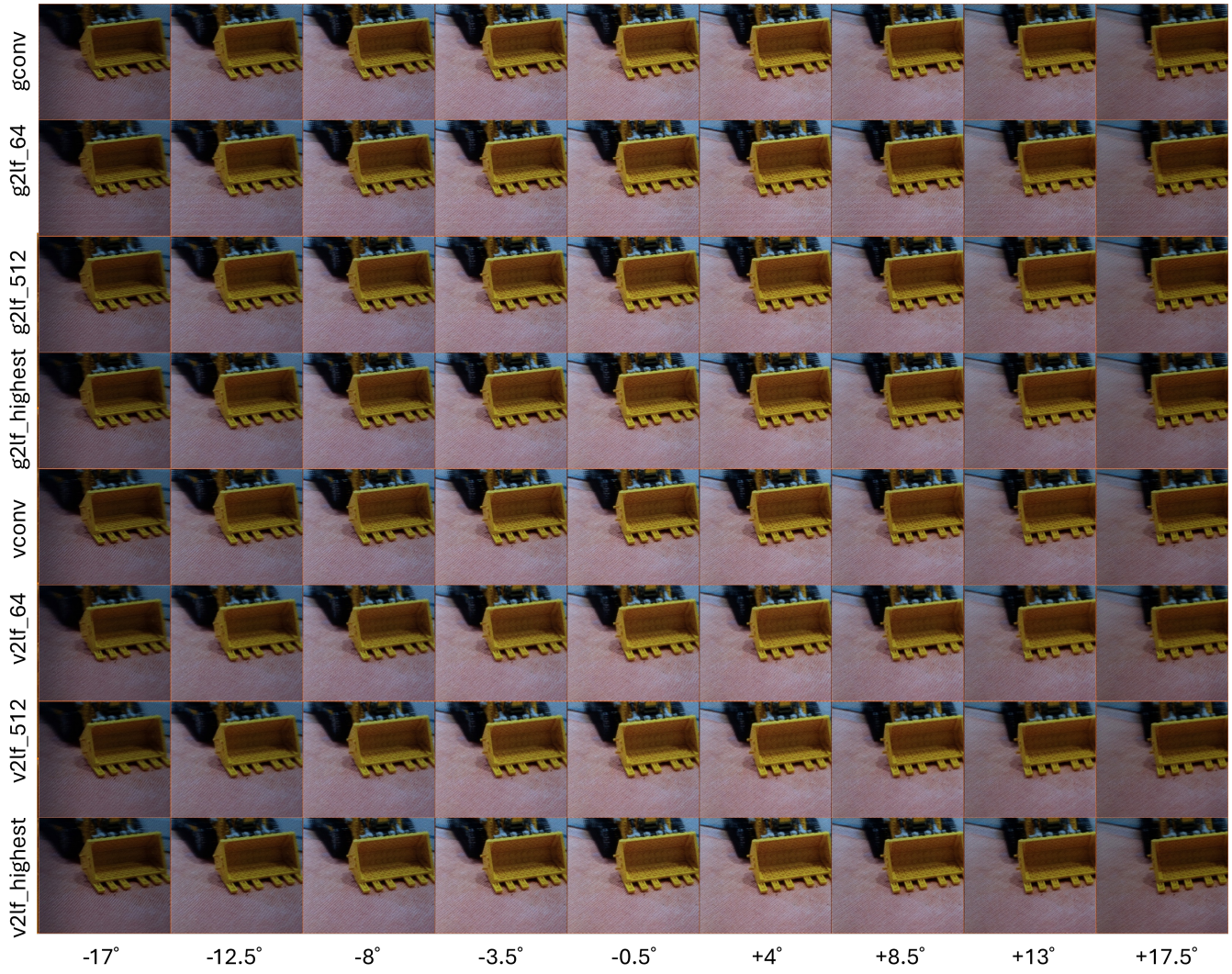

Fig. 14. Captured results of KITCHEN scene from multiple algorithm variants on a Looking Glass Go display, demonstrating accurate multi-view consistency and high-quality 3D reconstruction under real light field display conditions. Artifacts in individual views are less noticeable on the display, as multiple views perceptually blend within the viewer's pupil. More captured results are shown in the supplementary materials.