Light Field Display Point Rendering

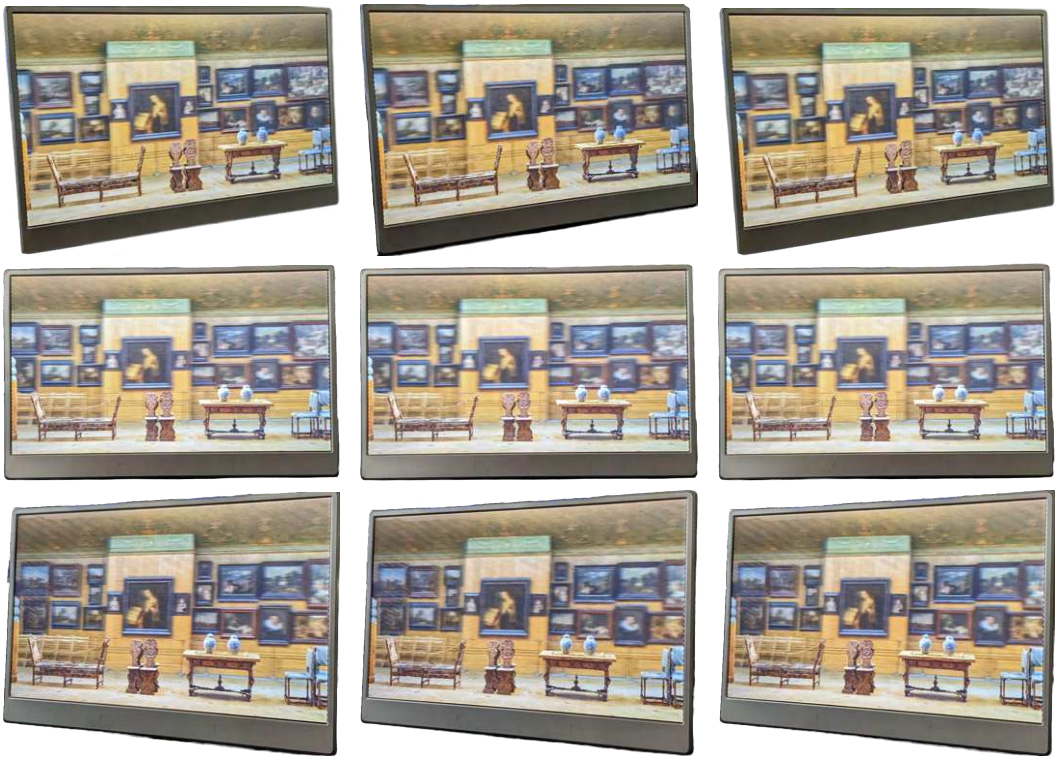


Fig. 1. *Results generated by our gold standard (GStd) (first column), light field display point rendering (LFDPR) (middle column), and standard multiview rendering (MVR) (last column) as viewed through the lens from left (top row), center (middle row), and right (bottom row) viewing angles. Each renderer generates 48 distinct views. While they generate very similar imagery, LFDPR makes imagery up to 8× faster than MVR.*

Rendering for light field displays (LFDs) requires rendering of dozens or hundreds of views, which must then be combined into a single image on the display, making real-time LFD rendering extremely difficult. We introduce light field display point rendering (LFDPR), which meets these challenges by improving eye-based point rendering [12] with texture-based splatting, which avoids oversampling of triangles mapped to only a few texels; and with LFD-biased sampling, which adjusts horizontal and vertical triangle sampling to match the sampling of the LFD itself. To improve image quality, we introduce multiview mipmapping, which reduces texture aliasing even though compute shaders do not support hardware mipmapping. We also introduce angular supersampling and reconstruction to combat LFD view aliasing and crosstalk. The resulting LFDPR is 2-8× times faster than multiview rendering, with similar comparable quality.

CCS Concepts: • **Computing methodologies** → **Rendering**; *Graphics processors*; *Point-based models*; • **Hardware** → *Displays and imagers.*

Author's address:

1 INTRODUCTION

In emerging light field displays (LFDs), each pixel emits several different colors, one for each of several angles of view, supporting unique views for stereoscopy and multiple simultaneous viewers, and simulating the experience of looking out (and around the edges) of a window. LFDs require multiview rendering to support angular variation in the light field they create.

Graphics hardware is primarily optimized for rendering scenes from a single perspective, making multiview rendering challenging. view independent rendering (VIR), improved view independent rendering (iVIR), and eye-based point rendering (EPR) [11, 12, 24] avoid multiple passes by generating and rendering points in real time. Frame by frame, they convert the model's triangles into a set of points fit to the views the current frame needs. With the points, they then render views in parallel, requiring nearly an order of magnitude fewer passes.

In this paper, we describe how we marshal and improve on the techniques of both iVIR and EPR to speed rendering for LFDs further. To achieve this, our light field display point rendering (LFDPR) includes the following innovations:

- *LFD-biased sampling* matches triangle sampling bias to LFD display sampling bias, reducing the size of the point cloud and improving efficiency.
- *Texture-based splatting* samples triangles coarsely when the textures mapped to them are also coarse, again reducing the point cloud and improving rendering speed.
- *Multiview mipmapping* supports texture antialiasing without requiring (currently unsupported) use of hardware mipmaps in the compute shader. This improves image quality with only a small impact on render speed.

These innovations enable rendering of LFD views 2–8× faster than standard multiview rendering (MVR), with nearly the same — and sometimes better — image quality.

2 RELATED WORK

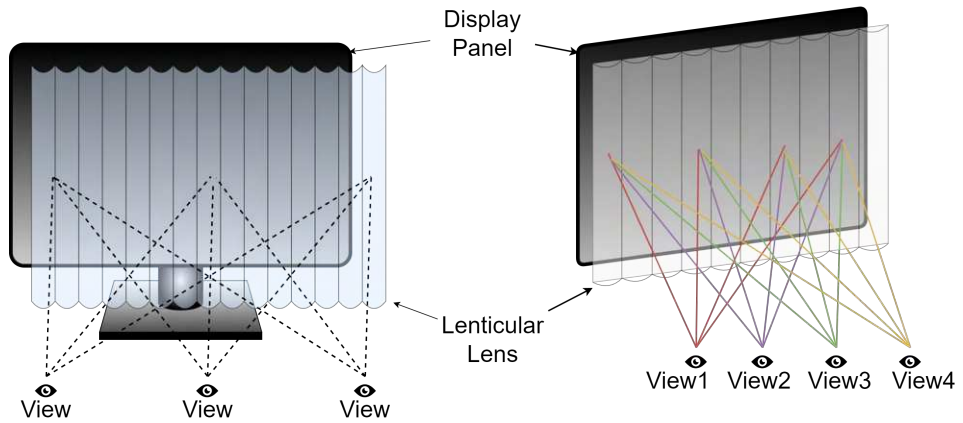


Fig. 2. *Viewpoints distributed across an LFD horizontally see different pixels and views through the lenticular lens. Each view corresponds to one panel pixel under each lenslet, creating a tradeoff between angular and spatial resolution.*

Light field rendering aims to capture and reproduce all views of a scene, including not only how color varies across space, but also across angle. At a given moment, the light field is radiance defined as a function of position and direction in an illuminated scene [13]. Light field displays show

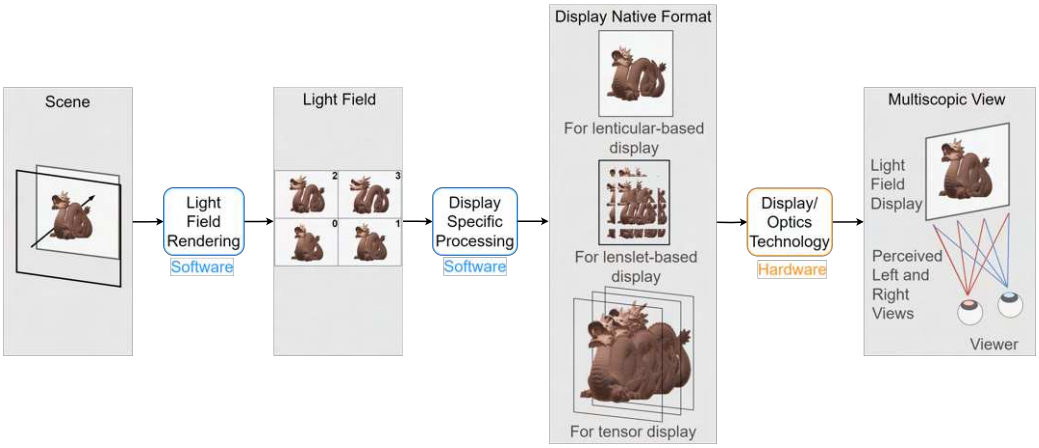


Fig. 3. *Conventional image formation process for light field displays. A 3D scene is processed to generate a fronto-parallel light field which is then processed to be compatible with the LFD. When viewed from the LFD, the displayed image is perceived as a 3D scene [9].*

the portion of the light field visible through the display surface, typically a rectangle. Light fields were first introduced to computer graphics in 1996 [14, 21]. Since then, a great deal of research has followed, including generation of light fields with machine learning [28, 38], and capture of them with cameras [25, 29, 40].

Our focus here is on LFDs. In most, each pixel represents not only the position and color of a light ray passing through the display surface, but also a range of orientations, with color varying across that range. Each pixel emits a different color based on the direction from which it is viewed, giving viewers different imagery as they move. Most often, to present views varying only horizontally, displays use parallax barriers that mask pixels not displaying the current view [19]; or a lenticular lens, an array of narrow cylindrical lenses that redirects the light rays emitted by pixels to the same end [26]. Both these types of LFDs offer stereoscopic and binocular depth (3D) cues, without requiring viewers to wear glasses or other technology. Fig. 2 demonstrates light flow for lenticular lens display systems. Note that such support for multiple views on a single display panel introduces a tradeoff between spatial and angular resolution.

LFDs supporting horizontal view change are now commercially available. Examples include products by FOVi3D, Dimenco and LeiaInc [2], Looking Glass Display [3], Light Field Lab [16] and AYE3D [1]. In particular, LeiaInc and Dimenco recently collaborated with ASUS and launched 16" light field display (LFD) laptops. LFDs are also finding their way into head-mounted displays (HMDs). NVIDIA proposed a near-eye LFD [20] that is light-weight and slim, incorporated into eyeglasses. These displays use the angular flexibility of light fields to eliminate the need for heavy, fixed lenses; to allow the use of HMDs without prescription eyeglasses; and to mitigate the vergence-accommodation conflict faced in most HMDs.

2.1 Light Field Rendering

Light field rendering is the process of generating synthetic light fields that form input for LFDs. Fig. 3 illustrates the full LFD image formation process; light field rendering is the first step it depicts.

To produce visual content for LFDs, it is crucial to capture the 3D spatial data of a scene from various viewing angles. This can be achieved by configuring a virtual camera array to capture a scene from multiple viewing directions. The resulting views are then combined into a format

compatible with a native LFD presentation [9]. One of the first examples is Levoy et al. [21], who simplified the 7D plenoptic function [4] (six degrees of angular and spatial freedom and radiance), into a 4D representation (a matrix of 2D images). This representation allowed computational efficiencies, uniform sampling of light fields, and control over the set of rays.

The standard approach to rendering light fields with GPUs is to make multiple passes over the scene description, with one pass per view (standard MVR). Because this is quite slow, Hübner et al. [18] proposed a single-pass solution that generates multiview splats and performs per-pixel ray-splat intersections in the fragment shader. In [17], they improved performance further with volume rendering method based on 3D textures. But their approach cannot directly render triangles and produces a maximum of eight views — not enough for modern LFDs. Sorbier et al. [8] duplicated geometry to reduce CPU-GPU bandwidth, rendering twice as fast as standard MVR for 80k triangles and 20 views. Our work also uses points to accelerate light field rendering with GPUs, but achieves much higher speedups.

Recently, AI has been employed to rapidly generate high-quality light fields within a specific viewing range for static scenes [10, 28, 38, 41] or even for scenes that have views changing over time [6, 36]. However, these approaches have limitations: they do not support scenes with moving objects, and were applied to scenes captured with cameras, rather than scenes rendered in real time. There is still considerable progress needed to achieve real-time rendering of high-quality content with AI-generated light fields.

2.2 Points, iVIR, and EPR

Using points as a rendering primitive can avoid many of the limitations of triangle rasterization [15, 22]. Unfortunately when views change, surface gaps can appear in projected point clouds. Preventing such artifacts often requires a prohibitive number of points, or similarly prohibitive filtering of sparser point sets (e.g., [34]). However, newer algorithms can avoid such painful tradeoffs. Ritschel et al. [32–34] adaptively sample according to brightness and view. Schutz et al. [35] apply compute shaders to render points more quickly.

Marrs et al.'s VIR [24] utilized the GPU rasterizer to generate frame-specific point clouds for off-screen views, enabling parallel rendering of views with significantly fewer geometry traversals than MVR. However, as views become more heterogeneous (as with environment maps or omnidirectional shadows), point cloud size grows, making real-time performance challenging. Gavane and Watson's iVIR [11] addresses this by optimizing triangle sampling and resizing off-screen buffers to match eye buffer resolution. This reduces the number of points and shader loads, enabling rendering of environment maps $2-4\times$ faster than MVR.

Gavane and Watson's EPR [12] further improved the performance and generality of point rendering for multiview effects. EPR tailors point clouds to the sampling rate required by the eye (frame) buffer instead of off-screen buffers, and by splatting resulting points across many off-screen pixels. EPR also deferred shading further by calculating off-screen lighting only when visible to the eye, and enabled very fast no-pass recursive reflection for dynamic environment mapping. As a result, EPR is $6-7\times$ faster than MVR and iVIR, with nearly identical image quality.

3 LFD POINT RENDERING

LFDPR combines iVIR's triangle sampling with EPR's point splatting to form a foundation for rendering LFD views. Fig. 4 shows an overview of the LFDPR pipeline. Like iVIR [11], the vertex shader checks vertex visibility, but uses views defined by the LFD rather than shadows or environment maps. Also like iVIR, the geometry shader adjusts sampling density, but to LFD view images with rectangular aspect ratios, and to match texture density (Section 3.1). Similar to EPR, LFDPR's fragment shader often splats points across multiple pixels when textures are coarser than the view

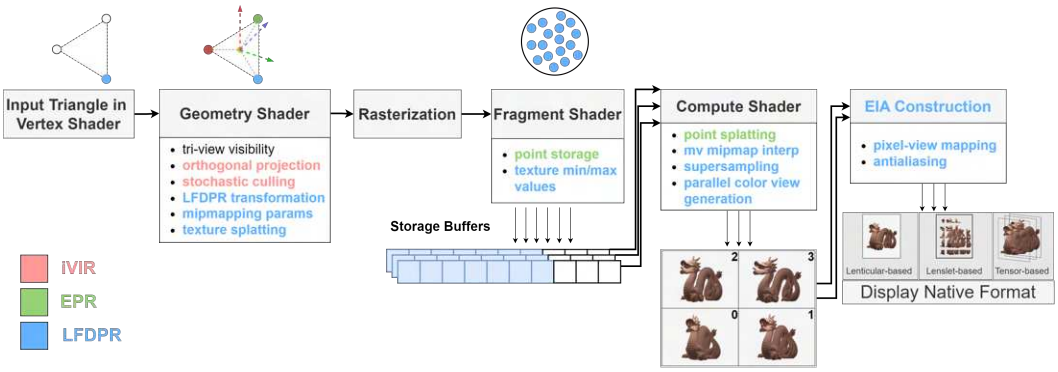


Fig. 4. *LFDPR pipeline as implemented in the GPU. Improvements of iVIR and EPR are marked in red and green font respectively, and LFDPR innovations are marked in blue font.*

buffer. However, it also implements multiview mipmapping (Section 3.3). Finally, a single image pass constructs the elemental image array (EIA), which interweaves rendered images for display on the panel, and viewing through the LFD's lenticular lens.

3.1 Determining Sampling Density

The geometry shader adjusts the point sampling density for each triangle using iVIR logic [11] by computing the projection matrix T_{align} applied to the triangle. However, LFDPR introduces three improvements: sampling according to LFD aspect ratio, texture-based sampling, and setup for multiview mipmapping.

With the angular-spatial tradeoff LFDs introduce (see Section 2), the individual views generated by LFD rendering often have rectangular aspect ratios, with more pixels in the vertical dimension. Furthermore, unlike off-screen views for shadows and reflections, the centers of projection for these views change predictably, being distributed horizontally across the LFD. Thus, triangles need not be sampled as densely in the horizontal as in the vertical direction.

To leverage this fact, LFDPR not only aligns triangles parallel to iVIR's view point-sampling plane, it also aligns the vertical and horizontal axes in iVIR's sampling view to the vertical and horizontal axes in the light field views. Thus T_{align} [11] not only orients the triangle's normal perpendicular to iVIR's view plane, but also maintains two sampling densities ($\vec{S}_{ortho}$) for the horizontal and vertical dimensions. To achieve this, instead of computing sampling density per unit area, we compute it horizontally and vertically by projecting pixel edges, see Fig. 5.

To make LFDPR sampling still more parsimonious, we introduce texture-based sampling. In iVIR and EPR, many points required to meet buffer sampling requirements may in fact be sampling the same texels, resulting in the same color. Texture-based sampling reduces point density to avoid this. In the geometry shader, we compute the maximum texel density on the triangle (denoted as $size_{uv}$) by dividing the area of triangle in the world space, A_{ws}, by the maximum number of texels n_{tex} a triangle spans over all the textures attached to it. This value is independent of any particular LFD view. Eq. 1 and Eq. 2 describe this process.

$$n_{tex} = \max_{t \in T} (A_{ts} \times X_t \times Y_t) \tag{1}$$

$$size_{uv} = \left(\frac{A_{ws}}{n_{tex}} \right) \tag{2}$$

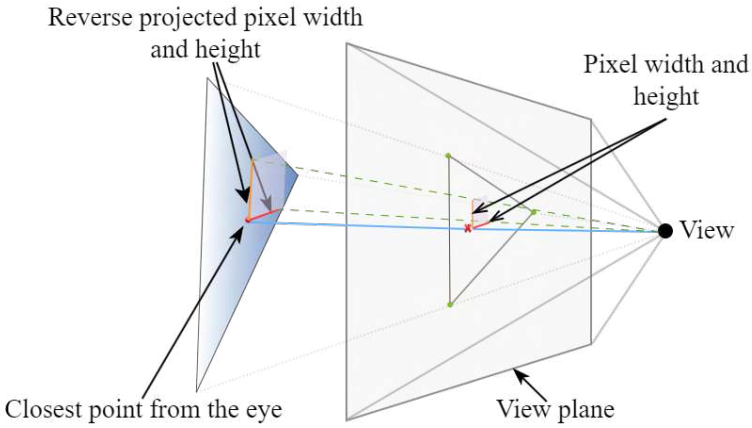


Fig. 5. *Reverse projection of pixel edges around the closest point on the polygon from the eye. The horizontal and vertical pixel edges often differ in length on LFDs.*

where A_{ts} is the area of triangle in texture space, and T is the set of all the textures attached to a triangle with each texture t of resolution $[X_t, Y_t]$. In iVIR, triangles are scaled by S_{ortho} to set the point sampling rate [11]. In LFDPR, when the $size_{uv}$ is greater than the size of the reverse projected view pixel $size_{xy}$, we adjust our sampling density using the formula given in Equation (3).

$$\vec{S}_{ortho} = \begin{cases} \vec{S}_{ortho} \times \sqrt{\frac{size_{xy}}{size_{uv}}} & \text{if } size_{uv} > size_{xy} \\ \vec{S}_{ortho} & \text{otherwise} \end{cases} \tag{3}$$

The geometry shader also sets up multiview mipmapping, discussed in Section 3.3.

3.2 Splatting

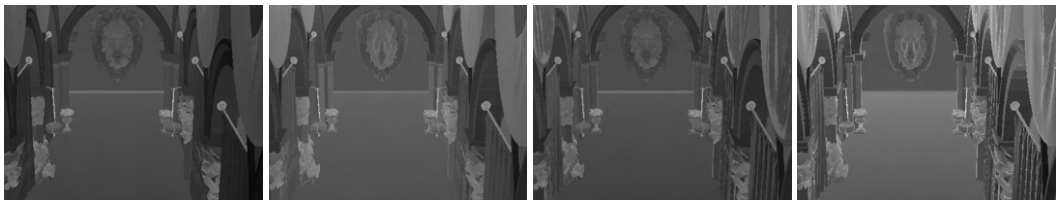
With texture-based sampling producing points sized larger than buffer pixels, we can no longer rely on iVIR's simple point projection. Instead, LFDPR uses EPR's splatting logic in the compute shader [12]. This not only avoids holes between points, but also improves image quality by enabling use of EPR's inside-triangle test during point splatting [12], resulting in near-rasterization edge quality. This is particularly important for light field rendering, when view and display are sampled at nearly the same rate, in contrast to shadow and reflection buffers.

3.3 Multiview Mipmapping

When texturing, mipmapping ensures good image quality by removing much of the aliasing that results when many texels project to the same view pixel. In iVIR and EPR, multiview point splatting in the compute shader does not support hardware mipmapping, but only shadows and reflections were rendered with points, leaving the majority of the eye view mipmapped. However when rendering light fields, every visible pixel is rendered with points, making the lack of mipmapping much more problematic. We introduce multiview mipmapping to address this problem.

When the geometry shader is processing a triangle, we compute both the minimum and maximum size any pixel on the triangle reaches in texture space across all views, represented as $size^{min}_{vpuv}$ and $size^{max}_{vpuv}$ respectively[1,2].

[1]When triangle textures do not share the same mapping, $size^{min}_{vpuv}$ and $size^{max}_{vpuv}$ will also depend on texture.
[2]These equations assume that geometric and texture space are proportionally isomorphic across the triangle, e.g. we assume that when X changes by 10%, so does U.



(a) *LFDPR Min LOD* **(b)** *LFDPR Max LOD* **(c)** *LFDPR Interp LOD* **(d)** *MVR LOD*

Fig. 6. *The proportion of an entire texture covered by pixels (LOD maps), as generated by LFDPR and MVR. LFDPR's multiview mipmapping generates two level of detail (LOD) values for each texture in each triangle: (a) the minimum LOD over all views, and (b) the maximum LOD over all views. In (c), a particular view's LOD map is produced by interpolating between the minimum and maximum LOD values. (d) presents the corresponding LOD map in MVR.*

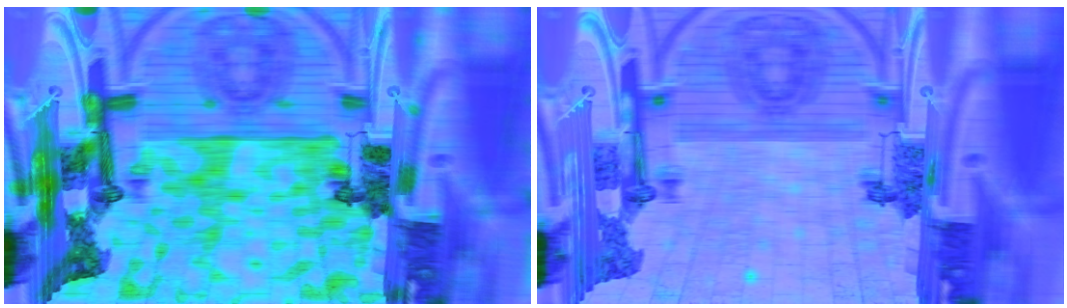
$$size^{min}_{vpuv} = \min_{v \in V} \left(size^{v}_{xy} \times \frac{A_{ts}}{A_{ws}} \right) \tag{4}$$

$$size^{max}_{vpuv} = \max_{v \in V} \left(size^{v}_{xy} \times \frac{A_{ts}}{A_{ws}} \right) \tag{5}$$

where $size^{v}_{xy}$ is the area of a pixel from view v projected into the triangle.

Eq. 6 and Eq. 7 show how the fragment shader transforms these size variables into the range typically used by graphics libraries such as OpenGL, often called level of detail or LOD. Given that each texture may have varying resolutions (e.g., albedo, normal, roughness, ao, metallic, etc.), the fragment shader calculates the minimum (LOD^{t}_{min}) and maximum (LOD^{t}_{max}) LOD values for each texture, uses those values to fetch minimum and maximum texture values from each texture's hardware mipmap (tv_{min} and tv_{max}), and stores those in the point buffer along with $size^{min}_{vpuv}$ and $size^{max}_{vpuv}$. Fig. 6a and Fig. 6b show min and max LOD maps for the sponza scene.

$$LOD^{t}_{min} = 0.5 \times log_2 \left(size^{min}_{vpuv} \times X_t \times Y_t \right) \tag{6}$$



(a) *LFDPR without mipmap* **(b)** *LFDPR with mipmap*

Fig. 7. *Perceptual quality of the EIA rendered by LFDPR (a) without multiview mipmapping and (b) with it. Here, HDR-VDP3 compares each image against our MVR gold standard, with blue indicating rarely perceived differences, green slightly more perceivable, and red usually seen.*

$$LOD^{t}_{max} = 0.5 \times log_2 \left(size^{max}_{vpuv} \times X_t \times Y_t \right) \tag{7}$$

During view generation in the compute shader, for each combination of view and point splat we interpolate a distinct displayed texture value tv in the range $[tv_{min}, tv_{max}]$, as described in Eq. 9. To perform this interpolation, we must first estimate the amount of texture the point splat covers in the current view, $size^{vs}_{uv}$. To do this, we take the area of the splat in texture space, and increase it in inverse proportion to the part of the view pixel the splat covers (we never decrease a splat's texture area, since we render per splat, not per pixel). We then clamp the area to the range $[size^{min}_{vpuv}, size^{max}_{vpuv}]$. In Eq. 8, $size^{vse}_{uv}$ is the estimated texture area adjusted by the proportion of the pixel it covers, $size^{vs}_{ndc}$ is the size of the splat in view space, in normalized device coordinates, and X_v and Y_v are the view resolution.

$$\begin{aligned} size^{vse}_{uv} &= \frac{size_{uv}}{\min(1, size^{vs}_{ndc} \times 0.25 \times X_v \times Y_v)} \\ size^{vs}_{uv} &= \text{clamp}(size^{vse}_{uv}, size^{min}_{vpuv}, size^{max}_{vpuv}) \end{aligned} \tag{8}$$

$$tv = tv_{min} + (tv_{max} - tv_{min}) \times \left(\frac{size^{vs}_{uv} - size^{min}_{vpuv}}{size^{max}_{vpuv} - size^{min}_{vpuv}} \right) \tag{9}$$

Fig. 6c shows the interpolation of the LOD in a view, and Fig. 7 shows the HDR-VDP3 perceptual quality map [23] comparing an LFDPR rendering of an EIA for the sponza scene with and without our mipmapping against the MVR gold standard (GStd) — a slowly rendered, very high quality image we use to evaluate image quality.

4 INTERLEAVING, RECONSTRUCTION, AND SUPERSAMPLING

Having rendered the necessary view images with LFDPR, we construct the EIA for the LFD by interleaving the views in a final image pass, as described in [9]. As we built these EIAs, we sometimes applied antialiasing and supersampling [5, 7]. While the benefits of these techniques in space are familiar, the value of their angular analogs are less well known. For reconstruction, we used Gaussian filters with a spatial diameter of 2.25 pixels, and an angular support of 2 view intervals. To integrate these filters, we sampled them 8 times using a random Gaussian distribution. We also investigated the value of supersampling, including modest 2× supersampling across space (view images with twice the resolution) and across angle (with splats projected into twice as many views, at random angles in view intervals). We discuss the benefits of this reconstruction and supersampling in Section 5.

5 RESULTS

In this section, we assess the effectiveness of LFDPR as measured by memory usage, rendering quality, and computation time. We begin by outlining the experimental setup we employed. Subsequently, we delve into rendering performance and image quality.

Scene (#tris)	Method	Performance				Error / Quality Measures for EIA (SV)				
		pt gen (#pts)	view gen	EIA constr	tot time	RMSE	P-det	Q	Q-JOD	SSIM
Sponza (1.0M)	MVR	-	30.74	8.18	38.92	1.81 (1.52)	0.74 (0.73)	9.84 (8.83)	9.95 (9.36)	0.91 (0.93)
	LFDPR no AA	2.81 (2.29M)	9.57	0.43	12.81 (3.04×)	2.30 (1.91)	0.34 (0.70)	9.66 (8.56)	9.87 (9.17)	0.86 (0.90)
	LFDPR Sp AA	2.78 (2.29M)	9.70	1.51	13.99 (2.78×)	2.07 (1.86)	0.32 (0.68)	9.75 (8.60)	9.91 (9.20)	0.89 (0.91)
	LFDPR View AA	2.72 (2.29M)	9.69	1.27	13.68 (2.84×)	2.29 (1.91)	0.34 (0.71)	9.65 (8.56)	9.87 (9.17)	0.87 (0.90)
	LFDPR Vi-Sp AA	2.73 (2.29M)	9.71	2.16	14.60 (2.66×)	2.07 (1.87)	0.32 (0.67)	9.75 (8.59)	9.91 (9.19)	0.88 (0.90)
Gallery (998.9K)	MVR	-	25.06	8.20	33.26	1.50 (1.23)	0.29 (0.75)	9.91 (9.09)	9.98 (9.54)	0.96 (0.97)
	LFDPR no AA	6.80 (2.60M)	12.17	0.43	19.40 (1.71×)	2.17 (1.92)	0.35 (0.74)	9.70 (8.42)	9.89 (9.07)	0.92 (0.93)
	LFDPR Sp AA	6.79 (2.60M)	12.15	1.48	20.42 (1.63×)	2.00 (1.87)	0.35 (0.73)	9.74 (8.46)	9.91 (9.10)	0.93 (0.93)
	LFDPR Vi AA	6.81 (2.60M)	12.16	1.25	20.22 (1.64×)	2.18 (1.92)	0.35 (0.74)	9.70 (8.42)	9.89 (9.07)	0.92 (0.93)
	LFDPR Vi-Sp AA	6.79 (2.60M)	12.16	2.14	21.09 (1.58×)	2.05 (1.87)	0.35 (0.73)	9.74 (8.46)	9.91 (9.10)	0.93 (0.93)
Coconut (2.0M)	MVR	-	44.45	8.23	52.68	0.52 (0.61)	0.16 (0.75)	9.98 (9.46)	9.99 (9.76)	0.98 (0.98)
	LFDPR no AA	16.53 (1.15M)	5.04	0.41	21.99 (2.40×)	0.59 (0.57)	0.17 (0.62)	9.97 (9.54)	9.99 (9.81)	0.98 (0.96)
	LFDPR Sp AA	16.55 (1.15M)	5.02	1.45	23.02 (2.29×)	0.51 (0.54)	0.13 (0.56)	9.99 (9.56)	9.98 (9.83)	0.98 (0.98)
	LFDPR Vi AA	16.61 (1.51M)	5.02	1.21	22.84 (2.31×)	0.57 (0.57)	0.16 (0.62)	9.97 (9.54)	9.99 (9.81)	0.98 (0.98)
	LFDPR Vi-Sp AA	16.62 (1.15M)	5.02	2.10	23.74 (2.22×)	0.50 (0.54)	0.12 (0.55)	9.98 (9.59)	9.99 (9.84)	0.98 (0.98)
Car (300.6K)	MVR	-	38.71	8.23	46.94	0.21 (0.33)	0.06 (0.72)	9.99 (9.73)	9.99 (9.90)	0.99 (0.99)
	LFDPR no AA	2.19 (669.4K)	3.22	0.42	5.82 (8.06×)	0.52 (0.57)	0.31 (0.83)	9.96 (9.53)	9.99 (9.80)	0.98 (0.98)
	LFDPR Sp AA	2.16 (669.4K)	3.21	1.44	6.81 (6.90×)	0.45 (0.54)	0.32 (0.82)	9.97 (9.57)	9.99 (9.83)	0.99 (0.99)
	LFDPR Vi AA	2.17 (669.4K)	3.22	1.21	6.59 (7.12×)	0.50 (0.57)	0.31 (0.83)	9.96 (9.53)	9.99 (9.80)	0.98 (0.98)
	LFDPR Vi-Sp AA	2.20 (669.4K)	3.22	2.09	7.51 (6.25×)	0.44 (0.54)	0.32 (0.80)	9.97 (9.58)	9.99 (9.83)	0.99 (0.99)

Table 1. *Rendering time and quality results for MVR and LFDPR across four models, as the reconstruction technique (AA or antialiasing) varied between none, spatial, view (angular) and view-spatial. No supersampling was used.*

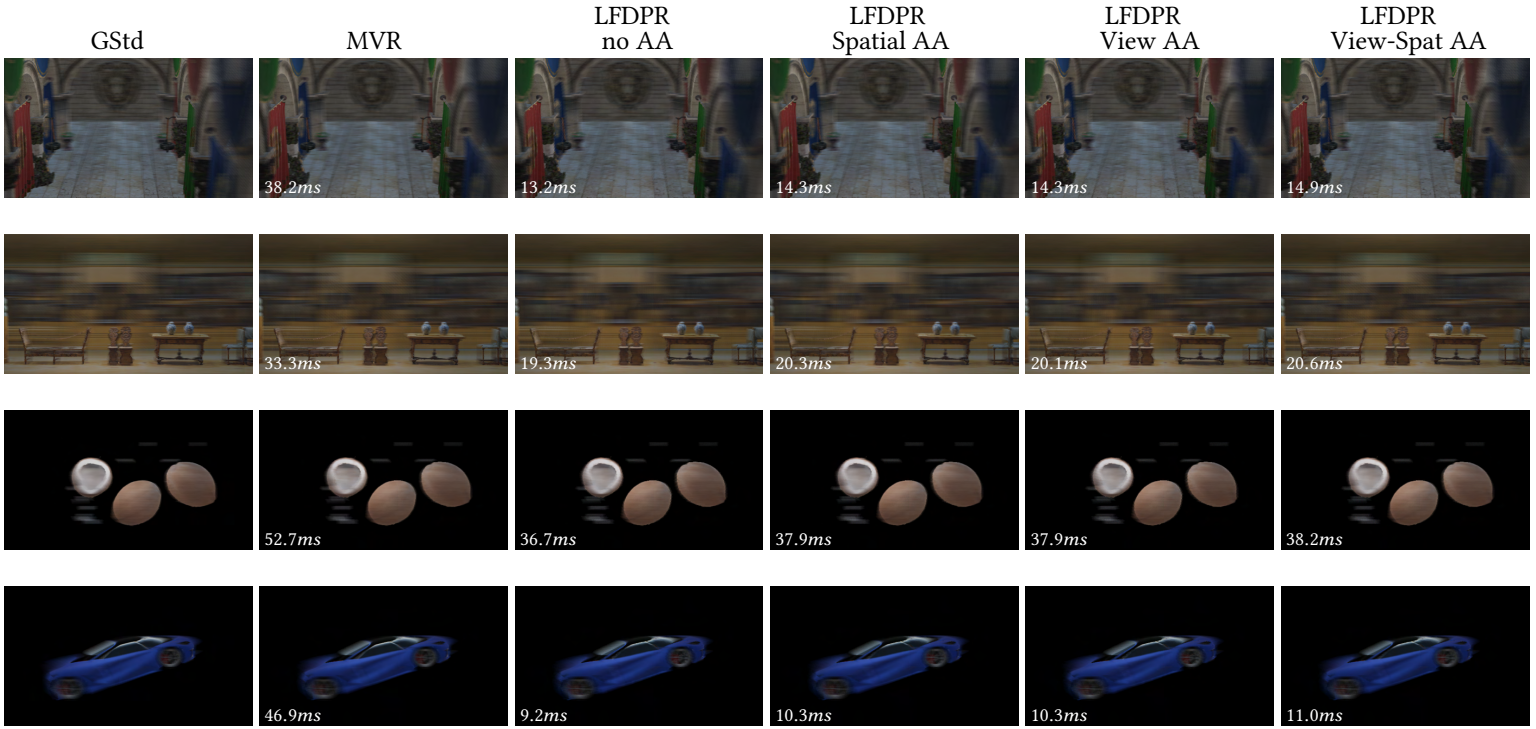


Fig. 8. *EIA generated using GStd, MVR, LFDPR without AA, LFDPR with spatial AA, LFDPR with view AA, and LFDPR with view and spatial AA; in the sponza, gallery, coconuts, and car scenes (from top to bottom). All rendering techniques generated 48 views (except GStd with 96 views). 5.*

5.1 Experimental Setup

We compared the light fields made by LFDPR and MVR. We used OpenGL 4.5 on a PC with an Intel i5-7600K @ 3.80 GHz CPU and an NVIDIA RTX 3070 GPU, running Windows 10. We rendered the sponza, gallery, car, and coconut scenes [27, 37] shown in Fig. 8. The sponza and gallery scenes are like many used in games, filling the entire view and making heavy use of textures. The gallery scene in particular has very large (16K×16K) textures. The coconuts do not fill the entire view, but still have a large number of triangles, and large textures. Finally the car has fewer triangles and no textures, but significant depth complexity (occluded components). All the scenes were dynamic, rotating around their centers, and used a physically-based rendering shader [31] that accessed albedo, roughness, metallic and normal textures. Our light field views used 64-bit unsigned integer buffers, with a resolution of 480×360. The MVR implementation rendered the light fields using multiple pipeline passes, each using hardware mipmapping. We compare all of our results to a slowly rendered, high quality MVR implementation with 96 4K resolution views using anisotropic mipmapping and Gaussian spatial reconstruction filters, sampled 32 times using a random Gaussian sampling scheme. We call this high quality image our gold standard (GStd). We displayed our rendering results on an LFD with a tilted lens [42], with a tilt angle of -9.66 degrees, 3840 x 2160 resolution, pixel width of 345.40 mm, height of 194.30 mm, lens count of 479.36, lens width of 0.72 mm, pixel pitch of 0.09 mm, and subpixel pitch of 0.03 mm.

5.2 Memory Comparison

In LFDPR, each view buffer required 1.32MB of memory; MVR's needed 2.63MB. LFDPR required 63.36MB for 48 view buffers and 80MB per million points; MVR required 126.24MB for 48 view buffers. Current GPU memories can contain more than a dozen GB.

5.3 Speed and Error Comparison

To compare performance, we averaged GPU run-time and the number of points generated over 1000 frames, with each technique generating the same views. Tables 1 and 2 compare LFDPR's speed to MVR's for the sponza, gallery, coconut, and car scene with no and 2× supersampling, respectively.

In Table 1, the leftmost and adjacent columns are the name of the scene and the method used to generate the imagery: MVR without anti-aliasing (AA) reconstruction; and LFDPR without AA, with spatial AA, with angular AA, and with spatial-angular AA. The next two columns show LFDPR's point generation time with the number of points in parentheses, and view generation time. The fifth and sixth columns report the EIA construction time and the total rendering time, along with the performance improvement as the ratio of LFDPR time over MVR's. The next five columns present the error and quality measures for both the EIA and a single view (in parentheses) compared against our gold standard. The first column of these five reports root mean-squared error (RMSE); the next three report quality measurements from HDR-VDP3 [23] including probability of detection *P-det*, quality correlate *Q*, and quality correlate in units of just objectionable differences *Q-JOD*; and the last is the structural similarity index measure (SSIM) [39]. RMSE ranges from 0 to 255, with lower values better; *P*-det is probability of detection ranging from 0 to 1, with lower values better; *Q* (quality correlate) ranges from 10 at best quality down to negative values; and lastly *Q-JOD* is similar to *Q* but scaled to JOD units, each corresponding to 75% of the population noticing the difference between the pair of images [30].

LFDPR renders these views up to 8× faster than MVR with comparable image quality. In the sponza scene, LFDPR's *P-det* was thrice as good as MVR's. LFDPR speed slowed as the number of triangles grew (see the coconuts), particularly when the number of texels per pixel was high

Scene (#tris)	SS	Method	Performance				Error / Quality Measures for EIA (SV)				
			pt gen (#pts)	view gen	EIA constr	tot time	RMSE	P-det	Q	Q-JOD	SSIM
Sponza (1.0M)	Spatial	MVR	-	33.33	8.21	41.54	1.43 (0.98)	0.34 (0.29)	9.92 (9.44)	9.98 (9.75)	0.94 (0.97)
		LFDPR Sp AA	5.14 (6.66M)	29.54	7.42	42.10 (0.99×)	1.68 (1.52)	0.29 (0.48)	9.89 (8.93)	9.97 (9.43)	0.92 (0.93)
	View	MVR	-	69.04	11.56	80.60	1.83 (1.52)	0.74 (0.73)	9.84 (8.82)	9.95 (9.36)	0.91 (0.93)
		LFDPR Vi AA	2.89 (2.37M)	21.33	7.13	31.34 (2.57×)	2.10 (1.90)	0.32 (0.67)	9.75 (8.58)	9.91 (9.19)	0.88 (0.90)
Gallery (998.9K)	Spatial	MVR	-	31.08	8.22	39.30	1.21 (0.89)	0.08 (0.54)	9.96 (9.42)	9.99 (9.74)	0.97 (0.98)
		LFDPR Sp AA	7.35 (5.57M)	22.26	7.26	36.86 (1.07×)	1.40 (1.32)	0.13 (0.54)	9.94 (8.99)	9.99 (9.47)	0.96 (0.96)
	View	MVR	-	49.51	12.39	61.90	1.38 (1.23)	0.29 (0.75)	9.91 (9.09)	9.98 (9.54)	0.96 (0.97)
		LFDPR Vi AA	7.01 (2.60M)	23.78	7.14	37.93 (1.63×)	2.01 (1.92)	0.34 (0.66)	9.75 (8.39)	9.91 (9.04)	0.93 (0.93)
Coconuts (2.0M)	Spatial	MVR	-	50.41	8.26	58.67	0.37 (0.31)	0.06 (0.37)	9.99 (9.85)	9.99 (9.95)	0.99 (0.99)
		LFDPR Sp AA	17.05 (1.16M)	5.27	6.98	29.30 (2.0×)	0.40 (0.42)	0.04 (0.34)	9.99 (9.75)	9.99 (9.91)	0.99 (0.99)
	View	MVR	-	89.07	10.83	99.91	0.50 (0.61)	0.14 (0.75)	9.98 (9.46)	9.99 (9.76)	0.98 (0.98)
		LFDPR Vi AA	16.88 (1.14M)	10.22	6.85	33.95 (2.94×)	0.51 (0.55)	0.12 (0.68)	9.98 (9.58)	9.99 (9.83)	0.98 (0.98)
Car (300.6K)	Spatial	MVR	-	46.22	8.21	54.44	0.21 (0.19)	0.04 (0.33)	9.99 (9.90)	9.99 (9.97)	0.99 (0.99)
		LFDPR Sp AA	2.60 (1.82M)	7.92	7.24	17.76 (3.07×)	0.32 (0.36)	0.05 (0.48)	9.99 (9.80)	9.99 (9.93)	0.99 (0.99)
	View	MVR	-	75.27	10.92	86.19	0.20 (0.33)	0.05 (0.72)	9.99 (9.73)	9.99 (9.90)	0.99 (0.99)
		LFDPR Vi AA	2.22 (669.4K)	7.02	7.21	16.45 (5.24×)	0.45 (0.54)	0.32 (0.81)	9.97 (9.56)	9.99 (9.82)	0.99 (0.99)

Table 2. *Results structured similarly to Table 1, but with all renderers using 2× supersampling. LFDPR did not use combined spatial-angular AA here.*

(coconuts and gallery). Our mid-range GPU has limited bandwidth, which began saturating when the number of points was particularly high (gallery). LFDPR's quality was always close to and sometimes exceeded MVR's quality.

Table 2 shows the results when using 2× supersampling in space (48 views, each with a resolution of 960 x 720) and views (96 views, each with a resolution of 480 x 360). To reach the displayed resolution, MVR performs a pixel-by-pixel unweighted average of the supersampled image. The leftmost column names the scene, and the adjacent two indicate either spatial or view supersampling, and the rendering method. The remaining columns follow Table 1's layout.

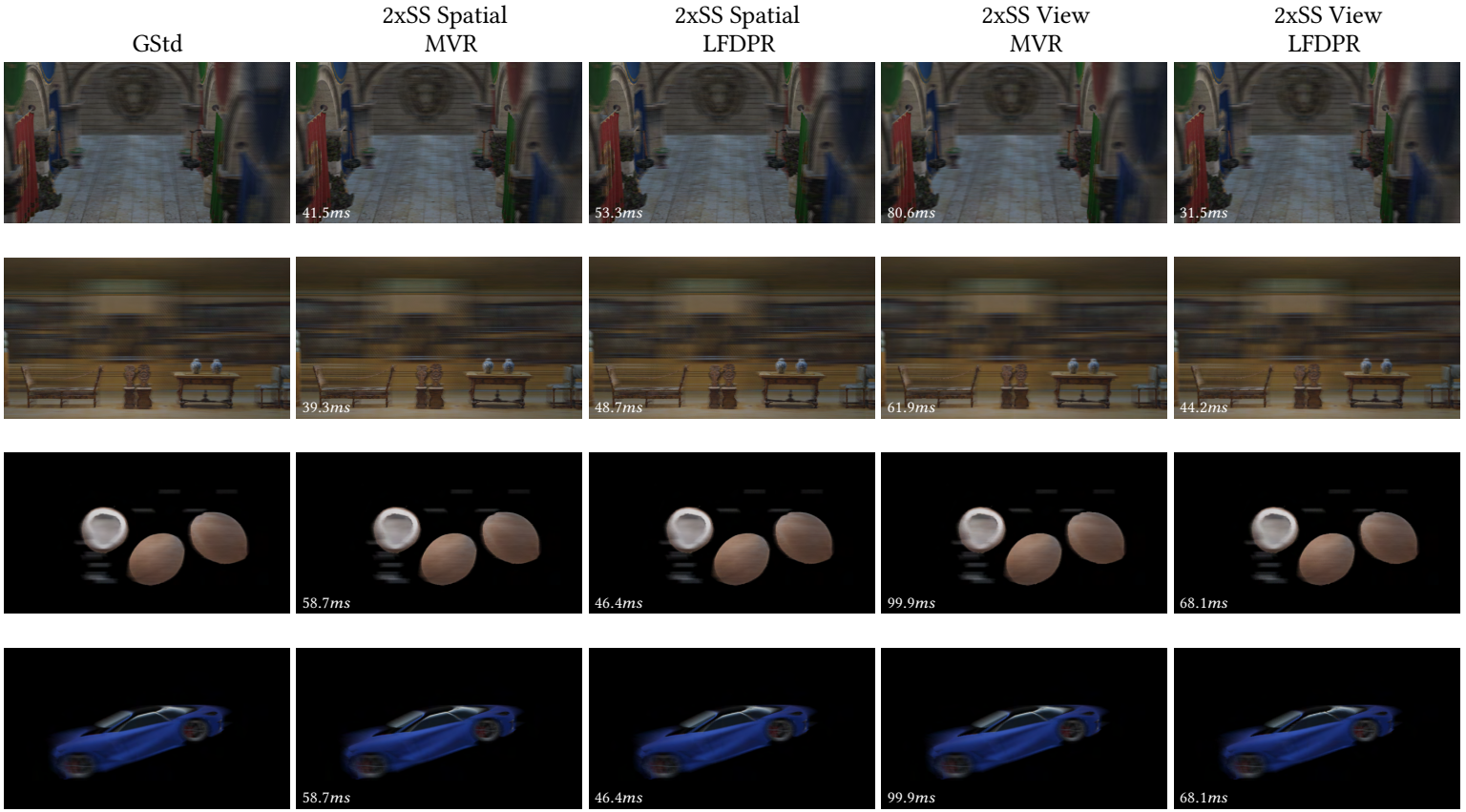


Fig. 9. *EIA generated using GStd, 2× spatial SS MVR and LFDPR with spatial AA, 2× view SS MVR and LFDPR with view AA; in the sponza, gallery, coconuts, and car scenes (from top to bottom). 2× spatial SS generates 48 views at a resolution of* 960×720*, and 2× view SS generates* 96 *views at a resolution of* 480×360*.*

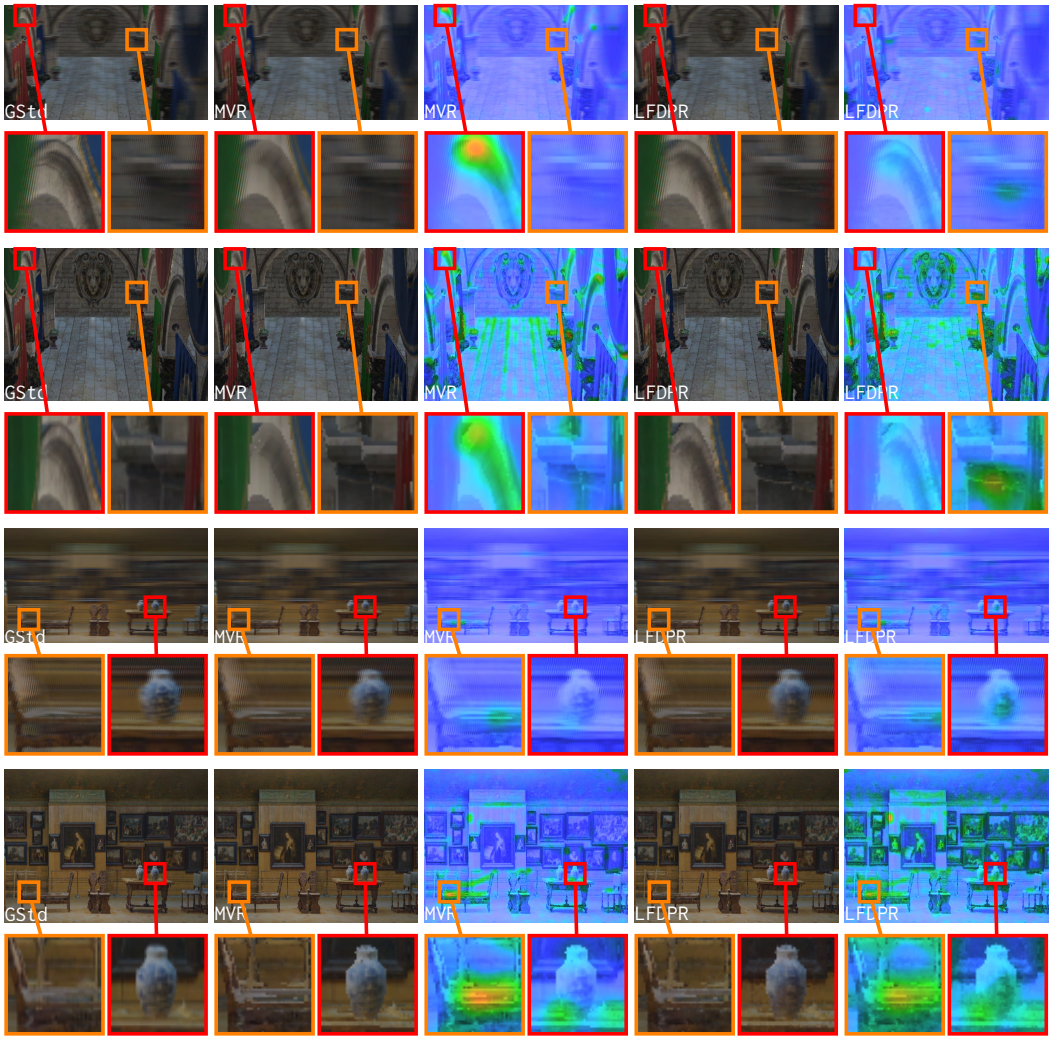


Fig. 10. *Quality comparison of GStd vs MVR and LFDPR with angular-spatial reconstruction. The leftmost column shows the GStd, the adjacent and central column show MVR and its HDR-VDP3 diifference image, and the right two columns show LFDPR and its difference image. The top four rows are the sponza, the bottom two the gallery.*

Here, LFDPR is up to 3× faster for spatial supersampling, and up to 5× faster for view supersampling, while maintaining nearly equivalent quality. In one case (spatial supersampling with the sponza), LFDPR became slightly slower than MVR. Note that spatial supersampling slows LFDPR more than angular, since filling two times more pixels can require roughly two times more points, while projecting the same set of points into random nearby views did not increase point cloud size. The reduction in rendering speed to achieve supersampling was not matched with a proportional improvement in quality. In general, spatial supersampling improved quality more than angular supersampling. However, angular supersampling is an essentially multiview technique, while our quality measures are single view techniques. We hope to assess the impact of angular supersampling more accurately using studies with human observers.

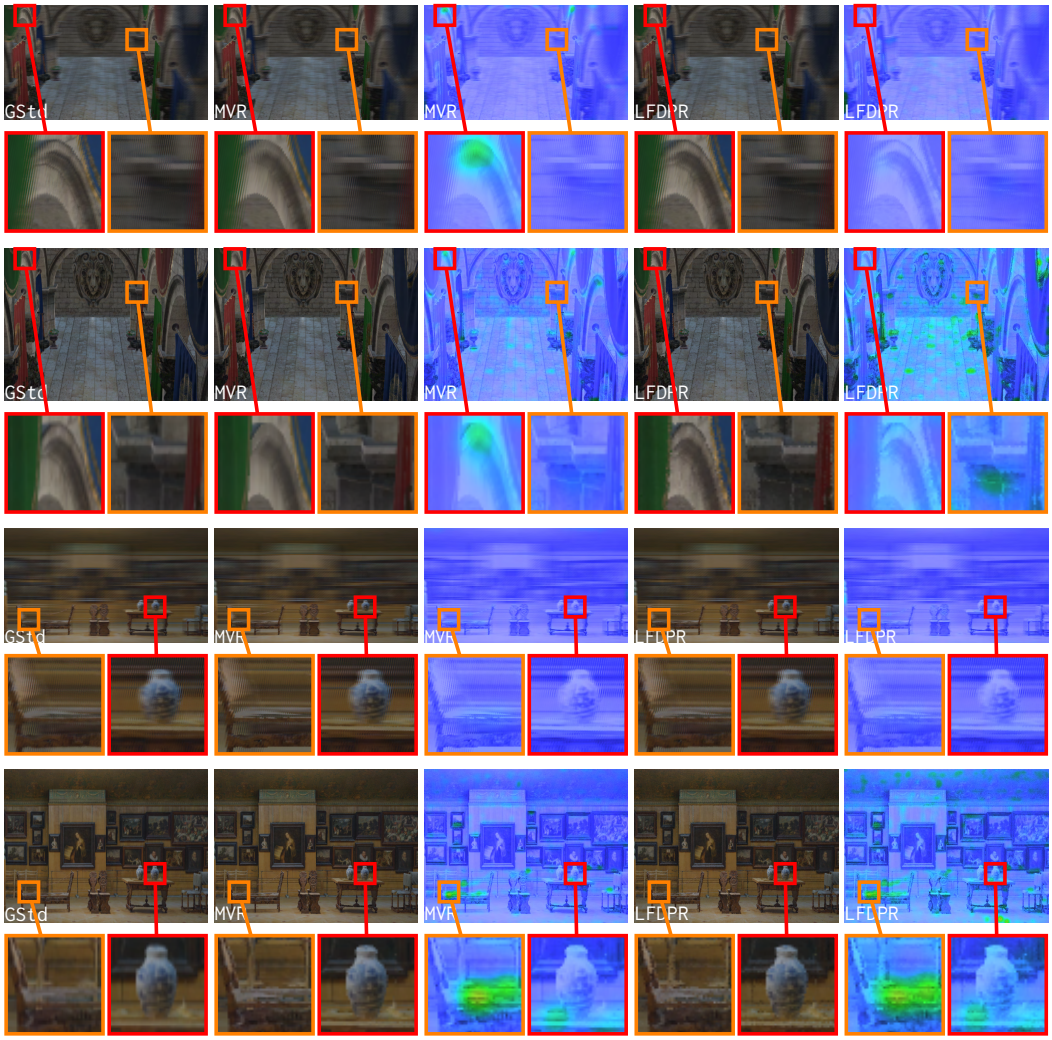


Fig. 11. *Quality comparison of GStd vs MVR and LFDPR with angular-spatial reconstruction, both with 2x spatial supersampling. The layout is the same as in Fig. 10.*

5.4 Quality Comparison

As we noted, while speed varied greatly, quality did not. Fig. 8 shows the differences in the sponza (top row), gallery (second), coconuts (third) and car (bottom row) as rendered by GStd (left column), MVR (second), LFDPR (third), LFDPR with spatial AA (fourth), with view AA (fifth), and view-spatial AA (rightmost column). The visual quality of LFDPR is quite comparable to MVR in all of these and LFDPR renders them much more quickly.

Fig. 10 zooms in on quality differences between MVR and LFDPR with angular-spatial reconstruction. Here, GStd is in the leftmost column, MVR and its HDR-VDP3 difference are in the second and third columns, and LFDPR is in the fourth and rightmost columns. LFDPR renders fine details in the sponza (top four rows), more clearly than MVR does. This can be seen clearly in the second row's red boxes. This may be due to LFDPR's flexible multiview mipmapping.

Fig. 9 is laid out similarly to Fig. 8, but uses 2× supersampling. These images look nearly identical to those in Fig. 8, despite being rendered much more slowly. Zooming in on the differences in Fig. 11 does not change this impression much, though the error images do reveal a reduction in error salience. Again, while supersampling did bring a modest improvement in quality, it came at a steep price in rendering speed. This tradeoff might be mitigated on a GPU with higher bandwidth buses.

6 CONCLUSIONS AND FUTURE WORK

Rendering for light field displays (LFDs) requires creating dozens or hundreds of views, which must then be combined into a single EIA image on the display. Multiview rendering at such scales is extremely challenging, making real-time LFD rendering extremely difficult.

Our innovations meet these challenges by marshaling and improving on the techniques of both iVIR and EPR. Unlike many multiview effects, all of the pixels generated for LFD rendering are (potentially) visible, making the eye-based efficiencies of EPR insignificant. We address these problems by introducing texture-based splatting, which avoids oversampling of triangles mapped to only a few texels; and with LFD-biased sampling, which adjusts horizontal and vertical triangle sampling to match the sampling of the LFD itself. LFD rendering's "every pixel" visibility not only makes high frame rates more difficult, it also makes higher quality imagery more difficult. To produce such quality, we introduce multiview mipmapping, which reduces texture aliasing even though compute shaders do not support hardware mipmapping; and use EPR's splatting, which avoids iVIR's edge bloat [11]. We also introduce angular supersampling and reconstruction to combat LFD view aliasing and crosstalk. The resulting LFDPR is 2-8× times faster than MVR, with similar comparable quality. Texture splatting and multiview mipmapping may also prove to be valuable in EPR.

In future work, we hope to examine LFDPR speed on higher bandwidth GPUs, which should make rendering with spatial supersampling much more performant. We also plan to study the effect on quality of angular reconstruction and supersampling in human experiments, since the error measures and single frame comparisons we present here do not capture the effects of view change well. In particular, we hope that these view angle quality improvements may reduce crosstalk, an LFD limitation in which LFD users see two images at the same time.

REFERENCES

[1] 2022. 3D Monitor without Galsses. https://aye3d.com/moniteur-3d/ [Online; accessed 01/04/2024].

[2] 2024. Dimenco - Products. https://www.dimenco.eu/sr-products [Online; accessed 01/04/2024].

[3] 2024. Documentation - Looking Glass Factory. https://docs.lookingglassfactory.com/ [Online; accessed 01/04/2024].

[4] Edward H Adelson, James R Bergen, et al. 1991. The plenoptic function and the elements of early vision. *Computational models of visual processing* 1, 2 (1991), 3–20.

[5] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. 2018. *Real-time rendering.* AK Peters/CRC Press.

[6] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2020. X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2020)* 39, 6 (2020). https://doi.org/10.1145/3414685.3417827

[7] Franklin C Crow. 1977. The aliasing problem in computer-generated shaded images. *Commun. ACM* 20, 11 (1977), 799–805.

[8] François De Sorbier, Vincent Nozick, and Hideo Saito. 2010. Gpu-based multi-view rendering. In *Computer Games, Multimedia and Allied Technology.* 7–13.

[9] Laura Fink, Svenja Strobel, Linus Franke, and Marc Stamminger. 2023. Efficient Rendering for Light Field Displays using Tailored Projective Mappings. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 1 (2023), 1–17.

[10] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. 2019. DeepView: View Synthesis With Learned Gradient Descent. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2362–2371. https://doi.org/10.1109/CVPR.2019.00247

[11] Ajinkya Gavane and Benjamin Watson. 2022. Improving View Independent Rendering for Multiview Effects. (2022).

[12] Ajinkya Gavane and Benjamin Watson. 2023. Eye-Based Point Rendering for Dynamic Multiview Effects. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 1 (2023), 1–16.

[13] Andreī Gershun. 1939. The light field. *J. Mathematics and Physics* 18, 1-4 (1939), 51–151.

[14] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 43–54.

[15] Markus Gross and Hanspeter Pfister. 2011. *Point-based graphics*. Elsevier.

[16] Jeff Heusser, John Montgomery, and Mike Seymour. 2018. Light Field Labs : 3D Holograms no glasses Deep Dive. https://www.fxguide.com/featured/light-field-labs-3d-holograms-no-glasses-deep-dive/

[17] Thomas Hübner and Renato Pajarola. 2007. Single-pass multi-view volume rendering. (2007).

[18] Thomas Hübner, Yanci Zhang, and Renato Pajarola. 2006. Multi-view point splatting. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. 285–294.

[19] Frederic E Ives. 1903. Parallax stereogram and process of making same. US Patent 725,567.

[20] Douglas Lanman and David Luebke. 2013. Near-eye light field displays. *ACM Trans. on Graphics* 32, 6 (2013), 220.

[21] Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proc. 23rd annual conference on Computer graphics and interactive techniques*. ACM, 31–42.

[22] Marc Levoy and Turner Whitted. 1985. *The use of points as display primitives (Technical Report TR 85-022)*.

[23] Rafal K Mantiuk, Dounia Hammou, and Param Hanji. 2023. HDR-VDP-3: A multi-metric for predicting image differences, quality and contrast distortions in high dynamic range and regular content. *arXiv preprint arXiv:2304.13625* (2023).

[24] Adam Marrs, Benjamin Watson, and Christopher G Healey. 2017. Real-Time View Independent Rasterization for Multi-View Rendering.. In *Eurographics (Short Papers)*. 17–20.

[25] Kshitij Marwah, Gordon Wetzstein, Yosuke Bando, and Ramesh Raskar. 2013. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

[26] Wojciech Matusik and Hanspeter Pfister. 2004. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 814–824.

[27] Morgan McGuire. 2017. Computer Graphics Archive. https://casual-effects.com/data

[28] 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.

[29] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. 2005. *Light field photography with a hand-held plenoptic camera*. Ph. D. Dissertation. Stanford university.

[30] Maria Perez-Ortiz, Aliaksei Mikhailiuk, Emin Zerman, Vedad Hulusic, Giuseppe Valenzise, and Rafał K Mantiuk. 2019. From pairwise comparisons and rating to a unified quality scale. *IEEE Transactions on Image Processing* 29 (2019), 1139–1151.

[31] Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

[32] Tobias Ritschel, Elmar Eisemann, Inwoo Ha, James DK Kim, and Hans-Peter Seidel. 2011. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2258–2269.

[33] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, H-P Seidel, Jan Kautz, and Carsten Dachsbacher. 2009. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 1–8.

[34] Tobias Ritschel, Thorsten Grosch, Min H Kim, H-P Seidel, Carsten Dachsbacher, and Jan Kautz. 2008. Imperfect shadow maps for efficient computation of indirect illumination. *ACM transactions on graphics (tog)* 27, 5 (2008), 1–8.

[35] Markus Schütz, Bernhard Kerbl, and Michael Wimmer. 2021. Rendering Point Clouds with Compute Shaders and Vertex Order Optimization. *arXiv preprint arXiv:2104.07526* (2021).

[36] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. 2022. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8269–8279.

[37] Colin Behrens Svenja Strobel. 2020. Coconut Scene.

[38] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

[39] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[40] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. 2005. High performance imaging using large camera arrays. In *ACM SIGGRAPH 2005 Papers*. 765–776.

[41] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. 2021. NeX: Real-time View Synthesis with Neural Basis Expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] Xunbo Yu, Xinzhu Sang, Duo Chen, Peng Wang, Xin Gao, Tianqi Zhao, Binbin Yan, Chongxiu Yu, Daxiong Xu, and Wenhua Dou. 2014. Autostereoscopic three-dimensional display with high dense views and the narrow structure pitch. *Chinese Optics Letters* 12, 6 (2014), 060008.