Check for updates

Received: 16 June 2025 | Revised: 5 November 2025 | Accepted: 12 December 2025

DOI: 10.4218/etrij.2025-0246

ORIGINAL ARTICLE

ETRI Journal WILEY

3D Gaussian rasterization for direct rendering of view-multiplexed images on multiview displays

Joonsoo Kim[1] | **Kugjin Yun**[1] | **Hyon-Gon Choo**[1] | **Hyeongju Mun**[2] | **Kyeongbo Kong**[2]

[1]Immersive Media Research Section, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

[2]Department of Electronics Engineering, Pusan National University, Busan, Republic of Korea

Correspondence
Joonsoo Kim, Immersive Media Research Section, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: joonsookim@etri.re.kr

Funding information
Institute for Information and Communications Technology Promotion, Grant/Award Number: RS-2022-II220022; National Research Foundation of Korea, Grant/Award Number: RS-2024-00414230

Abstract

Real-time rendering of three-dimensional scenes is essential for interactive, realistic display systems. Recently, representations based on 3D Gaussian primitives have emerged as a promising approach for novel view synthesis, offering both photorealistic image quality and real-time rendering. However, extending such renderers to autostereoscopic multiview displays is not straightforward because each subpixel must be rendered from a different viewpoint, whereas conventional renderers exploit shared viewport information to achieve efficiency. In this paper, we propose a rasterization method for 3D Gaussian-based representations that enables efficient rendering for multiview displays. By leveraging prior knowledge of the target viewport layout, the proposed method achieves an approximately 11 times speedup compared with the baseline without compromising image quality. We analyze the rendering performance of the proposed method under various combinations of renderer configurations and 3D scene datasets, demonstrating its potential for real-time multiview rendering.

KEYWORDS

3D Gaussian splatting, autostereoscopic 3D display, differentiable rasterizer, lenticular lens array, volumetric rendering

1 | INTRODUCTION

Lenticular multiview displays are a type of autostereoscopic display capable of providing depth cues to viewers, making the displayed images appear volumetric [1, 2]. This is achieved using cylindrical lenses that direct light emitted from adjacent subpixels toward different viewpoints. Leveraging the ability of multiview displays to present realistic 3D images, several telepresence systems incorporating these displays have been proposed in recent years. Lawrence et al. introduced the telepresence system called *Starline*, which enables 3D video conferencing with the sense of synchronous, face-to-face interaction [3]. While the *Starline* system synthesizes stereo images using a conventional depth-image-based rendering technique, other telepresence systems have utilized recent advances in novel view synthesis, such as radiance fields or 3D Gaussian splatting (3DGS) [4–8]. One such example is the system reported by Stengel et al., which performs 3D video conferencing using a single-camera video stream. In this system, monocular images are lifted into triplane-based representations that can be rendered

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (http://www.kogl.or.kr/info/licenseTypeEn.do).
1225-6463/$ © 2026 ETRI

from arbitrary viewpoints [9]. More recently, Tu et al. proposed a 3D telepresence system supporting resolutions up to 2K, with an encoder–decoder pair that transforms input images into 3D Gaussian representations at the core of the rendering pipeline [10].

To correctly display images on multiview displays, it is necessary to construct view-multiplexed images by interlacing multiple images corresponding to the target viewports [11]. The most straightforward approach involves rendering images separately for each viewport and then interleave the results. While this method is suitable for applications where real-time rendering is not critical, rendering separate images for each view introduces significant computational overhead, making it impractical for real-time applications [12, 13]. First, many rendered pixels are discarded during the interlacing process, as only those lying along narrow, slanted strips are visible through the lenticular lens array. Second, rendering multiple views on a regular grid leads to substantial redundancy, because many computations can be shared across viewports with minor architectural modifications. Consequently, various single-pass multiview rendering techniques have been proposed.

One such technique, introduced by Nozick et al., duplicates triangular primitives in the geometry shader based on their depth and the layout of the viewports, resulting in a tiled light field image [14]. Guan et al. proposed a custom rasterizer capable of processing multiview triangle arrays formed by sweeping triangular primitives horizontally in accordance with the viewport arrangement [15]. Both methods support efficient rendering across multiple viewports but are designed to generate complete rectangular images for general-purpose use and do not account for the specific structure of a lenticular lens array. In contrast, Fink et al. addressed the issue of pixel wastage by proposing a projection mapping to target pixels along slanted strips [16]. While this approach avoids rendering unnecessary pixels, it relies on multi-pass rendering and still requires a separate interlacing step after rendering each view individually.

Some prior works have also introduced single-pass renderers that directly produce view-multiplexed images without an explicit interlacing step. One such system, proposed by Hübner et al., uses a planar disk splatting-based renderer that calculates splat weights for each subpixel based on its corresponding viewport position [12]. For volumetric scene representations using 3D Gaussian kernels, Yang et al. introduced *DirectL*, which reformulates the rendering process from tile-based rasterization to ray-based rendering in order to maximize subpixel-level parallelism [13]. However, even with bounding volume hierarchy (BVH)-based acceleration, this shift to ray-casting led to a substantial decrease in rendering speed.

In this paper, we propose a view-multiplexed image renderer adapted to 3DGS. Unlike *DirectL*, our renderer retains the tile-based rasterization architecture, resulting in a more modest performance drop. Compared with Hübner's method, where the computational workload is concentrated in either the vertex or fragment shader, the proposed approach balances the workload across rendering stages by introducing shared variables across viewports.

2 | PROPOSED METHOD

2.1 | The overview of the modified Gaussian rasterizer

In this subsection, we describe the key modification on the 3D Gaussian rasterizer so that view-multiplexed images can be rendered. Following the setting of 3DGS, we assume the volumetric representation based on the 3D Gaussian kernel [5, 17]. Regarding the target multiview display, we consider one based on lenticular lens array that is capable of providing one-dimensional parallax. Soft three-dimensional ellipsoids, or simply splats, are defined by their colors, center positions, and variance and serve as the primitive elements such that an arbitrary three-dimensional scene of interest can be expressed as a combination of such soft ellipsoids. In the 3D Gaussian rasterizer proposed by Kerbl et al., each thread renders its corresponding pixel by considering the relative position between the pixel and the projected ellipsoidal image of the 3D Gaussian primitives. In order to render view-multiplexed image, it is necessary to consider projections of 3D Gaussian primitives for multiple viewports because each subpixel should be rendered as if it were a subpixel of its corresponding viewport. Specifically, a subpixel at (x,y) is assigned the index of a viewport using the following formula [11, 18]:

$$\gamma(x,y) = \frac{x - x_0 - y\tan\alpha}{h} \bmod 1, \tag{1}$$

where h, α, x_0 are the characteristic values of a multiview display. Because each thread can find out the normalized viewport number γ from the index of the thread, we modified the CUDA-based rasterizer so that the pixel-shading threads are aware of their view numbers.

In order to make threads aware of its corresponding viewports, it is necessary to take into account for the change of the projected ellipsoidal images across viewports. Figure 1A illustrates a case in which mth 3D Gaussian primitive is projected on a viewport other than the central viewport. When the viewport position is

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

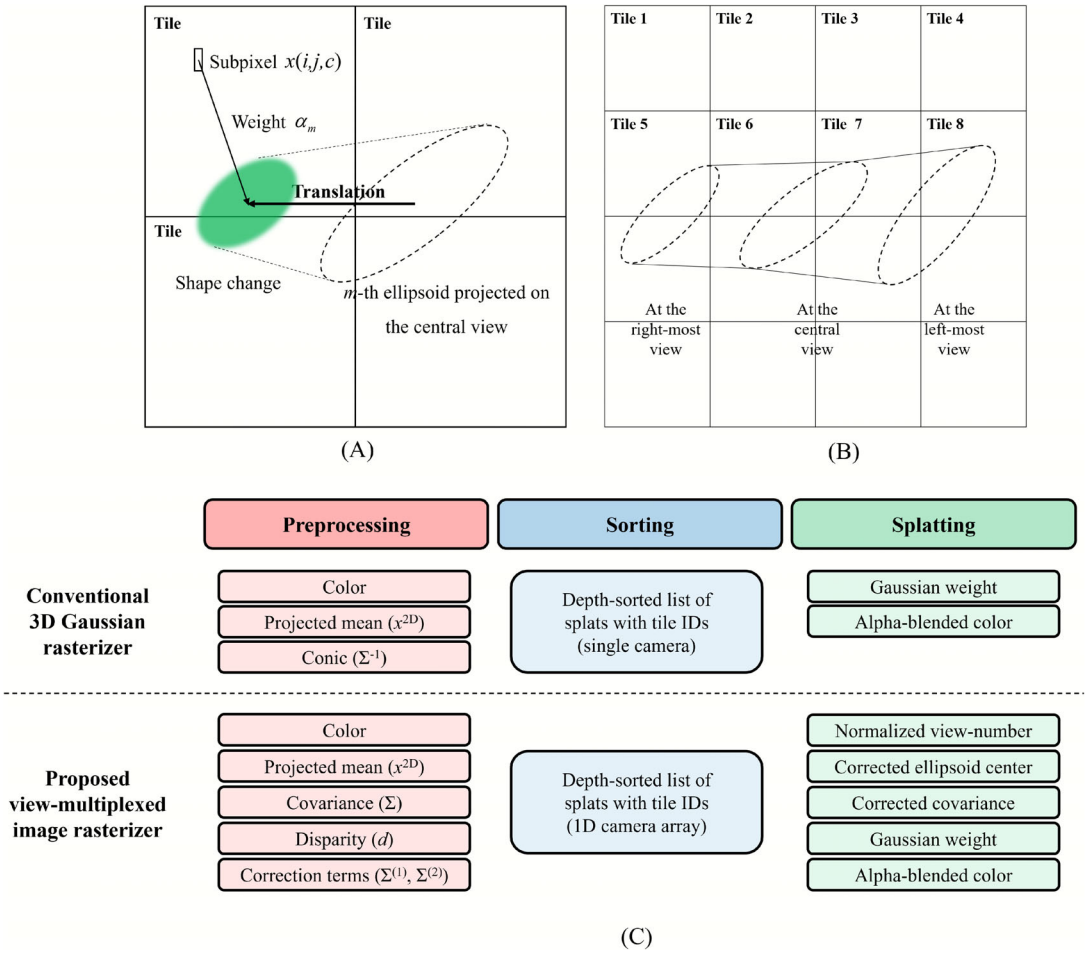


FIGURE 1 (A) Illustration of the shape change for a projected ellipsoidal image due to the shift of viewport camera. (B) Visualization of the overlap between the tiles and the projected ellipsoidal images for multiple viewports. (C) Comparison between the conventional 3D Gaussian rasterizer and the proposed rasterizer. The boxes on the top demonstrate the three major stages of the rendering, and the boxes below show the notable data obtained in each stage.

displaced from the central one, the projected ellipsoidal image undergoes a deformation. Because all the viewport cameras are parallel and horizontally aligned, the deformation can be decomposed into a horizontal translation and a shape change that can be represented by a 2×2 matrix. We will show in the next subsection that the amount of deformation can be computed by the pixel-shading threads using only simple operations.

In addition, we modified the bounding box estimation for projected ellipsoidal images. In the 3D Gaussian rasterizer, the threads for rendering a tile in an image form a block, and the block share the list of relevant 3D Gaussian primitives where a primitive is considered relevant if the bounding box of its projected image overlaps with the tile. Due to the deformation, the list of relevant primitives for a tile should list up any primitive that overlaps with the tile at least once in the course of deformation. For instance, in Figure 1B, even if the ellipsoid does not overlap with the fifth tile at the central and the left-most viewport, the primitive should be considered as relevant to the fifth tile because it overlaps with the tile at the right-most viewport.

Figure 1C shows the comparison of the conventional 3D Gaussian rasterizer and the proposed view-multiplexed image rasterizer. As in the conventional 3D Gaussian rasterizer, in the preprocessing stage, the colors $\{c_m\}$, the projected positions $\{x_m^{2D}\}$, and the conic matrices $\{\Sigma_m^{-1}\}$ of the projected splats are calculated for a given viewport. Then, moving on to the sorting stage, the overlaps between projected ellipsoids and the tiles are examined in order to list the Gaussian primitives that contribute to each tile. In the splatting stage, the pixel values $C(i,j,c)$ are computed in parallel by alpha-blending the contributions of the relevant Gaussian primitives identified in the sorting stage. To modify the rasterizer to render view-multiplexed images, pixel-shading threads additionally compute their corresponding view numbers and the deformations of the projected

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

ellipsoids in the splatting stage. In the preprocessing stage, intermediate variables such as disparity $\{d\}$ are calculated and stored for efficient computation of the deformations. Lastly, the modified bounding box estimation is applied to the sorting stage so that each tile can collect relevant 3D Gaussian primitives considering all the target viewports of the multiview display.

2.2 | Efficient projection of 3D Gaussian primitives to 1D array cameras

Because subpixels in tiles can be assigned different view numbers, they should be rendered with respect to different viewports in order to render view-multiplexed image. The most naïve way to accomplish this would be to recalculate the means x^{2D} and the variances Σ of the relevant primitives for every subpixel in the splatting stage. However, converting per-splat computation into a per-pixel computation generally has a negative impact on the computational efficiency; hence, it is desirable to maintain the roles of the preprocessing stage and reuse the results for multiple subpixels. Because all viewports are rectified, in the sense that they are oriented in parallel and horizontally aligned along a line, the projected means x^{2D} for two viewports differ only by horizontal disparity as follows:

$$x^{2D}_{\gamma+\Delta} = x^{2D}_{\gamma} + \Delta[d,0]^{T}, \tag{2}$$

where x^{2D}_{γ} and d denote the mean of the ellipsoidal primitive projected onto viewport γ and the disparity of the primitive across the whole viewing zone from $\gamma = 0$ to 1, respectively. Therefore, once one calculates the mean x^{2D} for a single viewport and the disparity d, it is possible to compensate x^{2D} for viewport changes by simply adding horizontal disparity $\Delta \times d$, where Δ is the amount of viewport change in the normalized view-number scale. This can be done in the splatting stage with a single additional addition operation per subpixel.

Similarly, it is possible to compensate the variance matrix Σ for viewport changes. Recall that the variance matrix can be calculated as follows [5]:

$$\Sigma = \mathbf{J}\mathbf{W}\Sigma^{3D}\mathbf{W}^{T}\mathbf{J}^{T}, \tag{3}$$

where $\Sigma^{3D} = \mathbf{R}\mathbf{S}\mathbf{S}^{T}\mathbf{R}^{T}$ is the 3×3 variance matrix defined by a scale matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$. $\mathbf{W}$ and $\mathbf{J}$ are the camera-world coordinate transformation matrix and the affine approximation of the projection mapping, respectively. Also, recall that the matrix $\mathbf{J}$ can be written as follows [17]:

$$\mathbf{J} = \begin{bmatrix} 1/z_0 & 0 & -x_0/z_0^2 \\ 0 & 1/z_0 & -y_0/z_0^2 \end{bmatrix}, \tag{4}$$

where x_0, y_0, z_0 are the coordinates in the viewport's camera coordinate system. Because we are considering a set of horizontally aligned cameras, the viewport change can be accounted for by modifying $\mathbf{J}$ as follows:

$$\mathbf{J}_{\gamma+\Delta} = \mathbf{J}_{\gamma} + \Delta\mathbf{J}^{(1)} = \mathbf{J}_{\gamma} + \Delta \begin{bmatrix} 0 & 0 & -I/z_0^2 \\ 0 & 0 & 0 \end{bmatrix}, \tag{5}$$

where I is the width of the viewing zone from $\gamma = 0$ to $\gamma = 1$. As a result, it is possible to compensate the variance matrix as follows:

$$\begin{aligned} \Sigma_{\gamma+\Delta} &= (\mathbf{J}_{\gamma} + \Delta\mathbf{J}^{(1)})\mathbf{W}\Sigma^{3D}\mathbf{W}^{T}(\mathbf{J}_{\gamma} + \Delta\mathbf{J}^{(1)})^{T} \\ &= \Sigma_{\gamma} + \Delta\Sigma^{(1)} + \Delta^{2}\Sigma^{(2)}. \end{aligned} \tag{6}$$

Therefore, the two correction terms $\Sigma^{(1)}$ and $\Sigma^{(2)}$ from the preprocessing stage can be stored and used to correct the variance matrix in the splatting stage. Note that (6) is not a low-order approximation but an exact compensation.

The above analysis shows that it is possible to compute the position and shape of the projected Gaussian primitive from those of another viewport. Leveraging this relation, we propose a method to compute the contribution of a Gaussian splat to a subpixel in a view-multiplexed image. First, we select the central viewport $\gamma = 0.5$ from the 1D camera array under consideration and compute the mean x^{2D} and the variance matrix Σ as in the preprocessing stage of conventional 3D Gaussian rasterizer. In addition, the disparity d and the correction matrices $\Sigma^{(1)}$ and $\Sigma^{(2)}$ are computed and stored for each Gaussian splat. When a thread calculates the contribution of mth splat for the subpixel at $x(i,j,c)$, the normalized view-number difference Δ with respect to the central view is computed. Then, we compensate x^{2D} and Σ using (2) and the (6) so that the alpha-blending weight can be calculated as follows:

$$\alpha_m = o_m e^{-(x - x^{2D}_{m,0.5} - \Delta d_m)^{T}(\Sigma_{m,0.5} + \Delta\Sigma^{(1)}_m + \Delta^{2}\Sigma^{(2)}_m)^{-1}(x - x^{2D}_{m,0.5} - \Delta d_m)}, \tag{7}$$

where $= x^{2D}_{m,0.5}$ and $\Sigma_{m,0.5}$ are the projected mean and variance matrix for the mth primitive at the central viewport. Note that (7) avoids computing the means x^{2D} and variances Σ from the 3D coordinates and the 3D variances of the splats for each subpixel, which would require several

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

matrix multiplications. Instead, a few linear combinations and a single 2×2 matrix inversion are required to modify the pixel value computation in the splatting stage so that view-multiplexed images can be rendered.

Before concluding this section, we note that the proposed method focuses on the geometric aspects of the projected Gaussian primitives and does not include a method to account for view-dependent color. Because there exist several representations, such as spherical harmonics or neural networks, to model view-dependent colors of the primitives, it is more challenging to construct an exact view-multiplexed image rendering algorithm without computing the color values of the primitives for each subpixel. However, we note that an approximate rendering algorithm can be proposed based on low-order expressions, because it suffices to approximate the view-dependent color along a finite line segment.

3 | EXPERIMENTAL RESULTS

3.1 | Implementation of the modified 3D Gaussian rasterizer

The modified 3D Gaussian rasterizer was implemented on top of the original differentiable rasterizer implementation by Kerbl et al. [5]. The followings are the major modifications made to the rasterizer:

- An additional parameter set that specifies the lenticular lens array parameters (h, α, x_0) and the width of the 1D camera array was passed to the rasterizer.
- In the kernel function *preprocessCUDA*, which corresponds to the preprocessing stage in Figure 1C, the disparity d and the correction matrices $\Sigma^{(1)}, \Sigma^{(2)}$ were additionally computed and stored for each Gaussian primitive. Also, the variance matrix was stored instead of the conic matrix, which is the inverse of the variance matrix. The data type *GeometryState* was also modified accordingly to accommodate the changes.
- The function *getRect*, which examines the overlap between projective primitives and tiles, was modified to receive the disparity as an additional input so that primitives visible from any of the viewports in the 1D camera array could be discovered.
- In the kernel function *renderCUDA*, which corresponds to the splatting stage in Figure 1C, the for-loop structure was modified so that alpha-blending was done on a per-subpixel basis. Also, the alpha-blending weight was calculated in accordance with (7), following the procedure described in Section 2.2.

In our implementation, the normalized view number γ is recomputed when rendering a new frame. This is essential if the lenticular lens array parameters vary over time, but in use cases where the parameters can be considered constant, the implementation can be modified to use a precomputed view-number table. Regarding the tile-ellipsoid overlap tests, the changes in both position and shape should be considered in principle, but unfortunately, it is not sufficient to examine only the two endpoints of the camera array. Therefore, to avoid calculating eigenvalues for the variance matrices at multiple viewports, we only take the disparity into account, which can be done without a significant increase in complexity. As can be seen later in Figure 2, this simplification does not result in noticeable artifacts near tile boundaries.

3.2 | Experimental settings

For the evaluation of the proposed method, the NeRF synthetic and Mip-NeRF 360 datasets were used [7, 19]. The NeRF synthetic dataset consists of image sets for eight virtual scenes, and the Mip-NeRF 360 dataset contains nine image sets for real scenes. The datasets were fed into the original 3DGS trainer to obtain the models for the 3D scenes. We exclude the "lego" and the "ship" scenes of the NeRF synthetic dataset because they required video memory over the capacity of the GPU in some of the benchmark tests. For the Mip-NeRF 360 dataset, the images were downscaled by a factor of four. We set the degree of the spherical harmonics to zero, but we note that this setting is not required to run the modified rasterizer. The remaining training hyperparameters were kept at their default values, with the number of training iterations set to 30 000. An NVIDIA RTX A6000 GPU was used for model training and rendering speed evaluation.

The width of the virtual 1D camera array was determined by considering the scale of the scene so that disparity ranges were normalized across different scenes. Specifically, we set the array width so that the disparity of the projections of the scene's center point between the two endpoints of the viewing zone corresponds to a certain proportion of the horizontal resolution. When we demonstrate the validity of the proposed method in Section 3.3, we set the disparity of the center point to roughly 15% of the horizontal resolution for clear visualization. Because this level of disparity is too large for practical use, we reduced the disparity range by a factor of five in the other experiments.

To verify the rendered view-multiplexed images on an actual multiview display, we used an off-the-shelf 3D display (Looking Glass 16-inch) with a resolution of

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

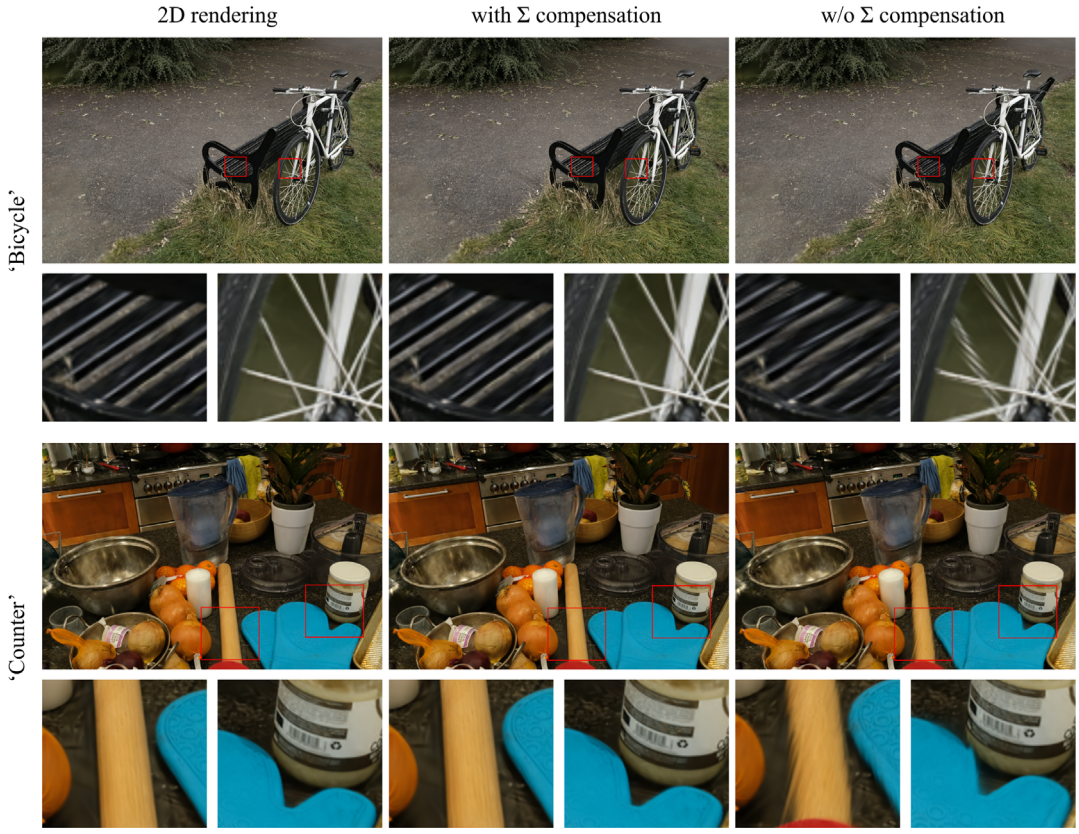


FIGURE 2 Comparison of the novel view images for the scenes "Bicycle" and "Counter" from the Mip-NeRF 360 dataset, rendered using the conventional 3D Gaussian rasterizer and the proposed rasterizer. Both rasterizers are set to render images for the left-most viewport of the virtual 1D camera array, where the central viewport of the array corresponds to one of the training viewports. The magnified regions, which highlight artifact suppression owing to the proposed variance matrix compensation (Σ compensation), are marked with red boxes.

3840×2160 [20]. The display was calibrated using the method described in [18], where the lenticular lens array parameters $h\tan$, α, and x_0 were measured to be 5.69 857 pixels, −0.131 902, and 5.94 788 pixels, respectively.

3.3 | Validity of the proposed rendering algorithm

Here, we show that the proposed method correctly renders pixels for any target viewport of the lenticular multi-view display. Figure 2 presents the rendered images for the left-most viewports of virtual 1D camera arrays, each including one of the training views as the central viewport. The images in the first column were obtained using the conventional 2D rasterizer, and those in the second column were obtained by setting the normalized view number γ to zero regardless of the subpixel coordinates in the modified rasterizer. As expected, the images that are rendered using different rasterizers are visually indistinguishable.

TABLE 1 Quantitative quality of the left-most viewport images using the projected ellipsoidal image compensation method described in Section 2.2.

	NeRF synthetic PSNR (dB)/SSIM/LPIPS	Mip-NeRF 360 PSNR (dB)/SSIM/LPIPS
GT/Reference	31.9029/0.9710/0.0283	27.3514/0.8226/0.1796
GT/Proposed	31.9023/0.9710/0.0283	27.2448/0.8226/0.1796
	RMSE	RMSE
Reference/Proposed	0.0002	0.0012

Note: "GT" refers to the ground-truth images of the test views, and "Reference" refers to the rendered images using the conventional rasterizer.

Table 1 shows the quality assessment results of the left-most viewport images when virtual 1D camera array positions are adjusted so that the left-most viewports are aligned with the test viewports of the datasets. Compared with the reference image rendered using the conventional rasterizer, the quantitative qualities of the left-most

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

viewport images using the proposed rasterizer with respect to the ground truth (GT) are almost the same as the reference images on average. The average difference between the images rendered using the two rasterizers measured in root mean square error (RMSE) has order of magnitude less than 10^{-3}.

The images in the third column of Figure 2 show the left-most viewport images using the modified rasterizer but with variance matrix Σ compensation bypassed. Compared with the case where variance matrix compensation is applied, recognizable artifacts appear, especially in regions where splats with large aspect ratios are present.

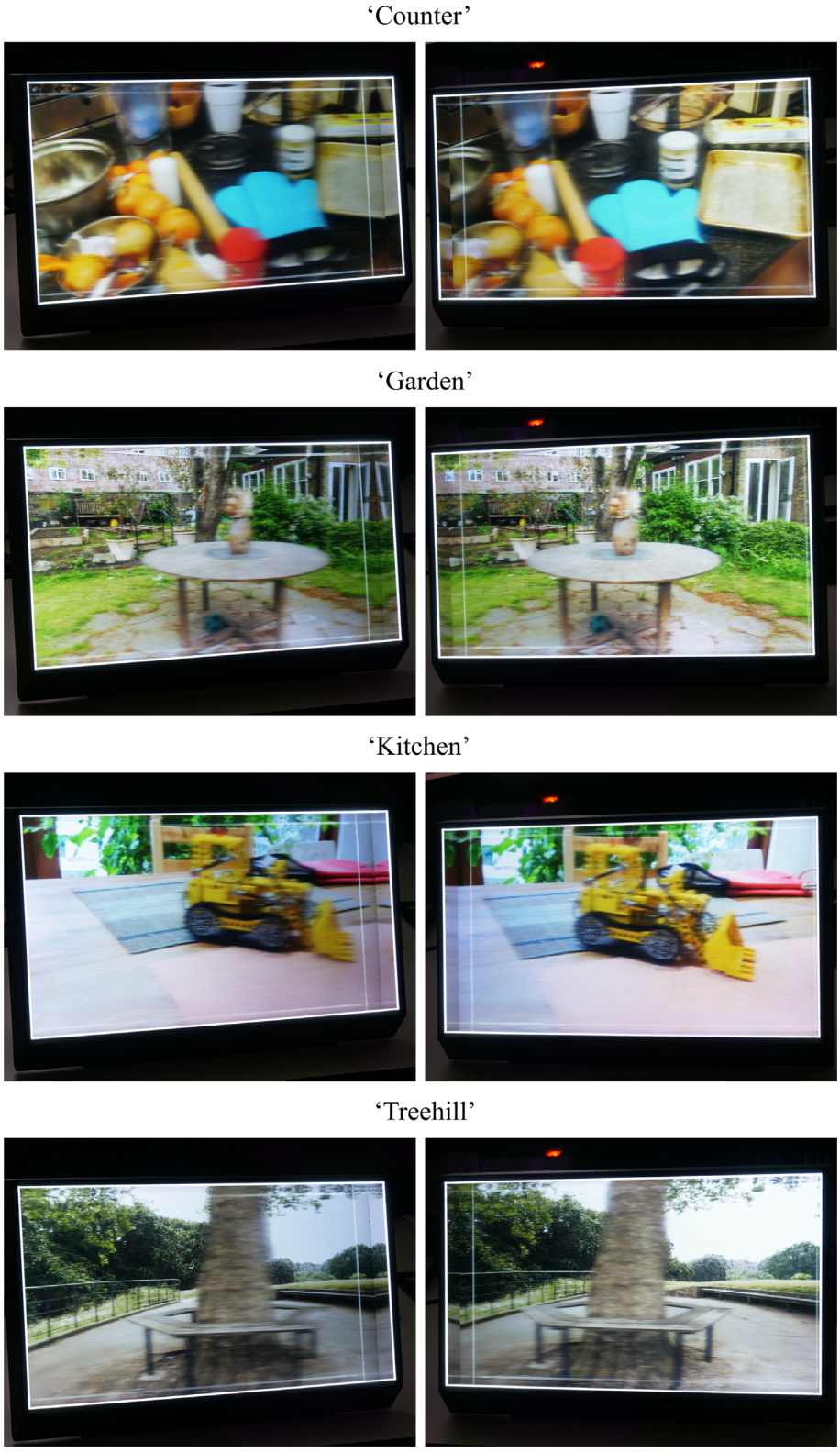


FIGURE 3 Visual results displayed on the 4K lenticular multiview display showing the scenes "Counter," "Garden," "Kitchen," and "Treehill" from the Mip-NeRF 360 dataset. The images were captured from two viewing positions using a mirrorless camera with a resolution of 5184 × 3888.

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

This observation suggests that the proposed method for compensating the variance matrices provides a clear benefit in terms of visual quality.

Figure 3 shows images of the 4K multiview display captured from two different viewpoints. Here, the normalized view number for each subpixel is calculated using (1), so that the view-multiplexed image can be rendered. When the images rendered by the modified rasterizer are displayed on the panel, motion parallax is correctly reproduced. Specifically, objects located closer to the viewer exhibit greater motion parallax, and that occluded regions in the background became visible as the viewing direction changes. Therefore, we conclude that subpixels are properly rendered by the modified rasterizer in accordance with their corresponding viewports, from which the subpixels are visible.

3.4 | Rendering speed of the proposed rasterizer

Now, we present the rendering speed of the proposed 3D Gaussian rasterizer. When choosing the virtual 1D camera arrays for rendering speed measurements, we took the test views of the datasets and set them as the central views of the camera arrays. Following the work by Yang et al. [13], we set the baseline by considering a scenario in which the conventional rasterizer is used to render individual viewports in a downscaled resolution and the final view-multiplexed image is obtained by spatially multiplexing the upscaled version of the rendered images. Specifically, low resolution (*LR*), medium resolution (*MR*), and high resolution (*HR*) are considered where the resolutions of the rendered images are 800×450, 1920×1080, and 3840×2160, respectively. Because the proposed rasterizer almost perfectly reproduces the view-multiplexed images in the *HR* scenario, we set the *HR* as the main baseline. The number of viewports in the virtual camera array is set at 60. Unlike in the work by Yang, we only measure the time to render 60 viewports and exclude the time for upscaling and spatial multiplexing. Instead of using multiple rasterizer instances to ensure full GPU utilization, we only use a single rasterizer instance and report the peak GPU utilization.

Table 2 shows the rendering speeds of the proposed rasterizer compared with the baseline. Compared with the *HR* baseline, the rendering speed of the proposed rasterizer is roughly 11 times faster. We note that *DirectL* achieved rendering speed improvement by the factor of 1.9 compared with the *HR* baseline [13]. Even if we consider the *LR* case and compensate the rendering speed by dividing them with the GPU utilization, the rendering speed of the proposed rasterizer is found to be 2.2 times faster. Therefore, the proposed method for rendering view-multiplexed images provides significant rendering speed gain without sacrificing the rendering quality.

3.5 | Ablation study

Table 3 summarizes the rendering speed of the rasterizer variants, where the variants are characterized by their constituent components. When either the component "Pixel mux." or "Subpixel mux." is used, the rasterizer renders view-multiplexed images. The view numbers are assigned per pixel and per subpixel for "Pixel mux." and "Subpixel mux.," respectively. Although pixel-level view multiplexing is generally not used because subpixels within the same pixel are often assigned significantly different view numbers, we include this case in our analysis to isolate the effect of separating alpha-blending loops for color channel. Regarding the components "Σ Comp." for variance matrix compensation, there are the proposed method described in Section 2.2 and a naïve method. When the naïve variance matrix compensation is used, the variance matrix is computed for every subpixel, starting from the 3D variance matrix Σ^{3D}. Lastly, we have the approximate matrix inversion component "Approx. mat. inv." When the component is used, the inverse of the variance matrix computed using the following approximation:

TABLE 2 Average rendering speed and GPU utilization of the proposed 3D Gaussian rasterizer in comparison with the baseline.

	NeRF synthetic		Mip-NeRF 360	
	Speed (fps)	**GPU util. (%)**	**Speed (fps)**	**GPU util. (%)**
LR (800×450, 60 views)	4.95	84.70	2.73	90.90
MR (1920×1080, 60 views)	2.87	94.30	1.53	89.00
HR (3840×2160, 60 views)	1.13	95.00	0.59	97.30
Proposed (3840×2160)	12.58	98.00	6.60	99.30

Note: In the baseline scenarios, the conventional 3D Gaussian rasterizer is used to render 60 viewports, separately.

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 3 Rendering speeds of the rasterizer variants depending on the components used.

	Components					Average rendering speed (fps)	
	Pixel mux.	Subpixel mux.	Σ Comp. (naïve)	Σ Comp. (proposed)	Approx. mat. inv.	NeRF synthetic (FHD/4K)	Mip-NeRF 360 (FHD/4K)
2D						160.87/67.38	90.67/36.11
3D-PX	✓					100.98/35.36	55.55/19.42
3D-P	✓					80.41/27.15	42.73/14.55
3D-SX		✓				61.53/20.29	31.89/10.76
3D-SN		✓	✓			11.16/3.47	5.59/1.77
3D-SA		✓		✓	✓	50.81/16.41	25.56/8.55
3D-S		✓		✓		39.86/12.58	20.24/6.60

$$\begin{aligned}&\left(\Sigma+\Delta\Sigma^{(1)}+\Delta^2\Sigma^{(2)}\right)^{-1}\approx\Sigma^{-1}+\Delta\sigma^{(1)}+\Delta^2\sigma^{(2)},\\&\sigma^{(1)}=-\Sigma^{-1}\Sigma^{(1)}\Sigma^{-1},\\&\sigma^{(2)}=\Sigma^{-1}\Sigma^{(1)}\Sigma^{-1}\Sigma^{(1)}\Sigma^{-1}-\Sigma^{-1}\Sigma^{(2)}\Sigma^{-1}.\end{aligned}\tag{8}$$

With the above approximation, it is possible to obtain an approximate inverse of the variance matrices without performing variance matrix inversions at the subpixel level. The rendering speeds of the variants are measured for 1920×1080 (FHD) and 3840×2160 (4K) resolutions.

Comparing the variant *3D-SX* against *2D*, the rendering speed is reduced by approximately a factor of three. The primary cause can be attributed to separate alpha-blending across color channels, although the increase in the number of relevant Gaussian primitives per tile also affects rendering speed. We observe that the rendering speed of the variant *3D-PX* is decreases by 37%-47% compared with the conventional rasterizer. The impact of variance matrix compensation on rendering speed can be evaluated by comparing *3D-PX* against *3D-P* and *3D-SX* against *3D-S*. When accurate variance matrix compensation is applied, the rendering speed drops by 20% to 39%. In contrast, when the approximate variance matrix correction is applied, the speed decreases more moderately by 17% to 20%.

Although the rasterizer for view-multiplexed image rendering is slower than its conventional counterpart, we emphasize that the proposed variance matrix compensation provides a significant speedup, as the variant *3D-S* is approximately 3.6 times faster than *3D-SN*. We also note that further content-adaptive optimization of the rendering pipeline is feasible. For instance, in tiles where all significant primitives are located sufficiently far away, pixel-level view multiplexing could be employed to enhance rendering performance.

3.6 | Rendering speed analysis under splat density control

To assess the potential of the proposed rasterizer for real-time rendering, we examined the dependence of rendering speed on the number of splats. Specifically, we prepared multiple 3D Gaussian representations for each scene, where the splat density was controlled by a threshold value n_{thres}. When the number of splats exceeded n_{thres} during training, densification was bypassed so that the number of splats could only decrease through pruning. Using the 3D Gaussian representations obtained without density control as the baseline, the threshold parameter was set proportional to the baseline number of splats. We selected 10 operating points such that the number of splats ranged from 10% to 100% of the baseline level. A decrease in splats counts results in image quality degradation, as shown in Figure 4A,C, where the PSNR typically drops up to 4 dB when the number of splats is reduced by a factor of 10.

Figure 4B,D shows the rendering speed of the variant *3D-S* at 4K resolution as a function of the number of splats used to represent the 3D scenes. 3D Gaussian representations with more splats tend to include smaller splats to capture high frequency details, and such splats are more likely to affect fewer pixels. Hence, the rendering speed decreases exponentially as the number of 3D Gaussian splats increases across both datasets. We observe that rendering speed can increase from 3.2–16 to 7.1–47 fps at the cost of PSNR drop of less than 1 dB. The results suggest that a proper trade-off between rendering quality and speed enables real-time rendering of view-multiplexed images. It is noteworthy that rendering quality can be adjusted either by controlling splat density or by selecting alternative rasterizer variants such as *3D-SX* or *3D-P*. Furthermore, if techniques such as 3D Gaussian compression or multi-scale representation are applied to

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

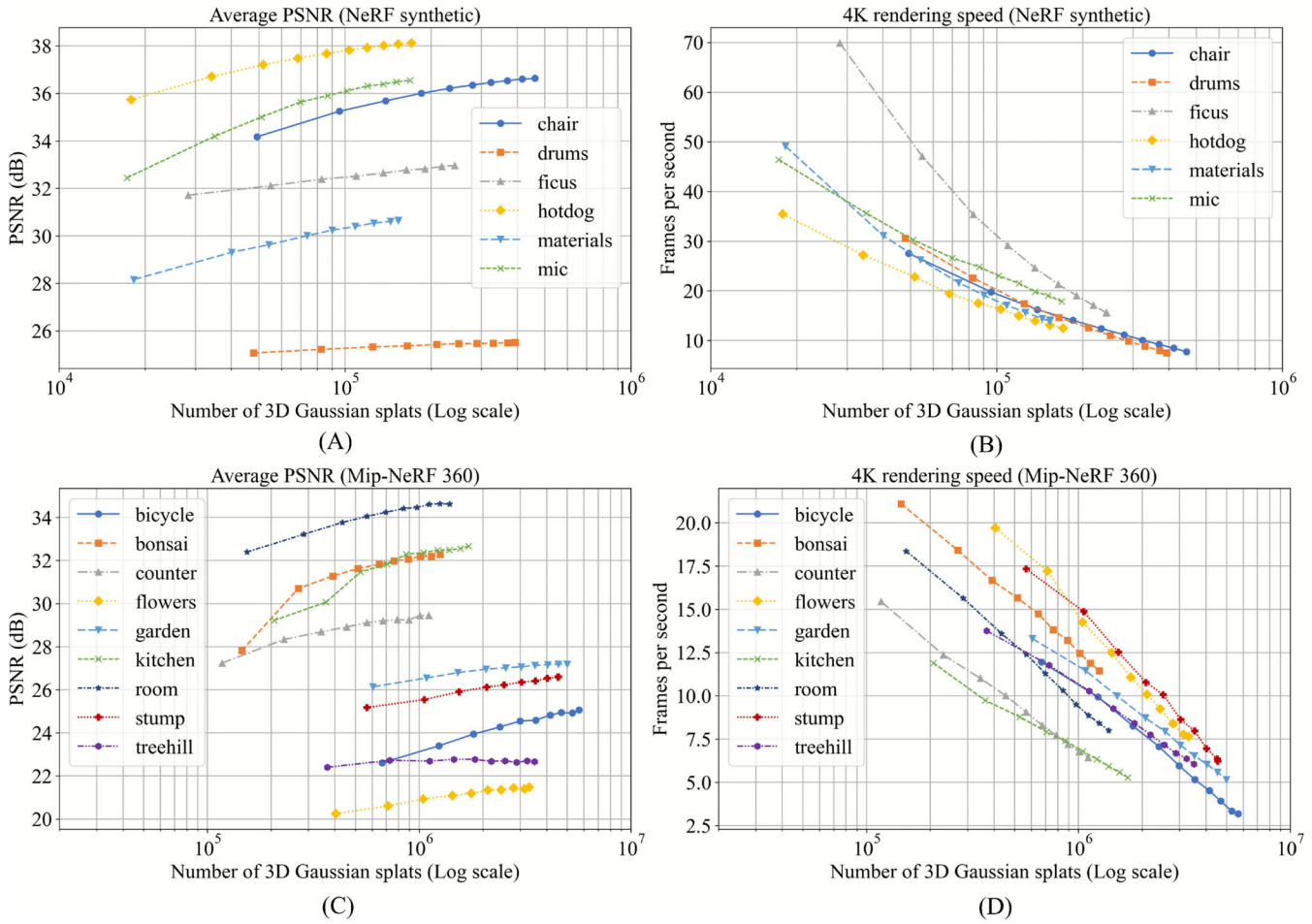


FIGURE 4 Rendering speed of the proposed renderer (*3D*) for the NeRF synthetic and Mip-NeRF 360 datasets as a function of the number of splats. The quality of the 3D scene representations at each operating point is evaluated by measuring the average peak-to-signal ratio (PSNR) for the test views. Each line plot, shown using 10 operating points, corresponds to a scene in the test datasets.

reduce the number of primitives, an even better trade-off between quality and speed can be expected [21, 22].

4 | CONCLUSION

In this paper, we proposed a rasterizer that renders view-multiplexed images from 3D Gaussian primitives. Leveraging the one-dimensional arrangement of target viewports assumed in lenticular multiview displays, our method computes splat projections only once for the central viewpoint, as projections for the remaining viewports can be derived through low-complexity transformations. The proposed rasterizer was implemented on top of a conventional 3D Gaussian rasterizer while preserving the tile-based rendering scheme. We showed that the proposed method for translating the means and variance matrices between viewports can closely reproduce images obtained by performing individual splat projections per view. Crucially, avoiding separate projections leads to significant rendering speed gain, approximately an 11 times improvement. The rasterizer correctly renders view-multiplexed images and achieves real-time performance at 4K resolution. Considering its compatibility with existing 3D Gaussian rasterization pipelines, we believe our rasterizer is a promising solution for real-time rendering on multiview displays.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Joonsoo Kim https://orcid.org/0000-0002-6470-0773
Hyon-Gon Choo https://orcid.org/0000-0002-0742-5429

REFERENCES

1. N. A. Dodgson, *3D without the glasses*, Nature **495** (2013), no. 7441, 316–317.

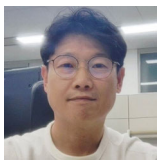
22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

2. B. Lee, S. Park, K. Hong, and J. Hong, *Design and implementation of autostereoscopic displays*, Bellingham, WA, USA, SPIE Press, 2016.
3. J. Lawrence, D. Goldman, S. Achar, G. M. Blascovich, J. G. Desloge, T. Fortes, E. M. Gomez, S. Häberling, H. Hoppe, A. Huibers, C. Knaus, B. Kuschak, R. Martin-Brualla, H. Nover, A. I. Russell, S. M. Seitz, and K. Tong, *Project Starline: a high-fidelity telepresence system*, ACM Trans. Graph. **40** (2021), no. 6, 242. DOI 10.1145/3478513.3480490
4. J. Choi, Y. Lee, H. Lee, H. Kwon, and D. Manocha, *MeshGS: adaptive mesh-aligned Gaussian splatting for high-quality rendering*, (Proc. Asian Conf. Comput. Vision), 2024, pp. 3310–3326. DOI 10.1007/978-981-96-0969-7_16
5. B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, *3D Gaussian splatting for real-time radiance field rendering*, ACM Trans. Graph. **42** (2023), no. 4, 1–139.
6. T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, *Scaffold-GS: structured 3D Gaussians for view-adaptive rendering*, (Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.), 2024, pp. 20654–20664. DOI 10.1109/CVPR52733.2024.01952
7. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *NeRF: representing scenes as neural radiance fields for view synthesis*, Commun. ACM **65** (2021), no. 1, 99–106.
8. Z. Yang, X. Gao, Y.-T. Sun, Y. Huang, X. Lyu, W. Zhou, S. Jiao, X. Qi, and X. Jin, *Spec-Gaussian: anisotropic view-dependent appearance for 3D Gaussian splatting*, Adv. Neural Inform. Process. Syst. **37** (2024), 61192–61216.
9. M. Stengel, K. Nagano, C. Liu, M. Chan, A. Trevithick, S. De Mello, J. Kim, and D. Luebke, *AI-mediated 3D video conferencing*, (ACM SIGGRAPH 2023 Emerging Technol., SIGGRAPH '23, Association for Computing Machinery, New York, NY, USA), 2023, pp. 1–2. DOI 10.1145/3588037.3595385
10. H. Tu, R. Shao, X. Dong, S. Zheng, H. Zhang, L. Chen, M. Wang, W. Li, S. Ma, S. Zhang, B. Zhou, and Y. Liu, *Tele-Aloha: a low-budget and high-authenticity telepresence system using sparse RGB cameras*, arXiv preprint (2024). DOI 10.48550/arXiv.2405.1486
11. C. V. Berkel, *Image preparation for 3D LCD*, (Stereoscopic Displays and Virtual Reality Systems VI, San Jose, CA, USA), *Stereoscopic displays and virtual reality systems VI* 1999, pp. 84–91.
12. T. Hübner, Y. Zhang, and R. Pajarola, *Multi-view point splatting*, (Proc. 4th Int. Conf. Comput. Graphics Interact. Tech. Australasia Southeast Asia, Association for Computing Machinery, Kuala Lumpur, Malaysia), 2006, pp. 285–294.
13. Z. Yang, B. Liu, Y. Song, L. Yi, Y. Xiong, Z. Zhang, and X. Yu, *DirectL: efficient radiance fields rendering for 3D light field displays*, ACM Trans. Graph. (TOG) **43** (2024), no. 6, 1–19.
14. V. Nozick, F. de Sorbier, and H. Saito, *Multi-view rendering using GPU for 3-D displays*, GSTF J. Comput. **1** (2010), no. 1, 44–49.
15. Y. Guan, X. Sang, S. Xing, Y. Chen, Y. Li, D. Chen, X. Yu, and B. Yan, *Parallel multi-view polygon rasterization for 3D light field display*, Opt. Express **28** (2020), no. 23, 34406–34421.
16. L. Fink, S. Strobel, L. Franke, and M. Stamminger, *Efficient rendering for light field displays using tailored projective mappings*, Proc. ACM Comput. Graphics Interact. Techniq. **6** (2023), no. 1, 1–17.
17. M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, *EWA volume splatting*, (Proc. Visualization, San Diego, CA, USA), 2001 pp. 29–538.
18. J. Kim, G. Lee, H. Eum, H.-C. Shin, Y. Park, and J. Seo, *Iterative calibration of a multiview 3D display with linear extrinsic crosstalk using camera feedback*, Appl. Opt. **57** (2018), no. 16, 4576–4582.
19. J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, *Mip-neRF 360: unbounded anti-aliased neural radiance fields*, (Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.), 2022, pp. 5470–5479. DOI 10.1109/CVPR52688.2022.00539
20. Looking Glass Factory, *Looking glass documentation*, Accessed 2025. 6. https://docs.lookingglassfactory.com/
21. Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, *LightGaussian: unbounded 3D Gaussian compression with 15x reduction and 200+ FPS*, Adv. Neural Inform. Process. Syst. **37** (2024), 140138–140158.
22. B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, *A hierarchical 3D Gaussian representation for real-time rendering of very large datasets*, ACM Trans. Graph. (TOG) **43** (2024), no. 4, 1–15.

AUTHOR BIOGRAPHIES



Joonsoo Kim received his B.S. and Ph.D. degrees in Electrical Engineering from Seoul National University (SNU), Seoul, Republic of Korea, in 2012 and 2017, respectively. He has been with the Electronics and Telecommunications Research Institute since 2017, where he is currently a senior member of the research staff in the Immersive Media Research Section. His current research interests include light field displays, autostereoscopic 3D displays, computer vision, and machine learning.



Kugjin Yun received the M.S. degree in Computer Engineering from Chungbuk National University, Cheongju, Republic of Korea, in 2001, and the Ph.D. degree in Electronics and Radio Engineering from Kyung Hee University, Seoul, Republic of Korea, in 2016. In 2001, he joined the Electronics and Telecommunications Research Institute. He is currently a principal member of the research staff with the Immersive Media Research Section, Media Research Division. His research

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

interests include augmented reality (AR), virtual reality (VR), light field (LF) media processing, machine learning, and realistic media technology.



Hyon-Gon Choo received the B.S. and M.S. degrees in Electronics Engineering in 1998 and 2000, respectively, and a Ph.D. degree in Electronic and Communication Engineering in 2005 from Hanyang University, Seoul, Republic of Korea. He is currently a principal researcher at the Electronics and Telecommunications Research Institute in Daejeon, Republic of Korea. He is the currently the director of the Immersive Media Research Section. His research interests include video coding for machines, holography, multimedia protection, and 3D broadcasting technologies.



Hyeongju Mun received the B.S. degree in Electrical and Electronics Engineering from Pusan National University, Busan, Republic of Korea, in 2025. He is currently pursuing an M.S. degree in Electrical and Electronics Engineering at Pusan National University, Busan, Republic of Korea. His current research interests include image processing, computer vision, machine learning, and deep learning.



Kyeongbo Kong received the B.S. degree in Electronics Engineering from Sogang University, Seoul, Republic of Korea, in 2015, and the M.S. and Ph.D. degrees in Electrical Engineering from the Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea, in 2017 and 2020, respectively. From 2020 to 2021, he worked as a postdoctoral fellow with the Department of Electrical Engineering, POSTECH, Pohang, Republic of Korea. From 2021 to 2023, he was an assistant professor in the Media School at Pukyong National University, Busan. He is currently an associate professor of Electrical and Electronics Engineering at Pusan National University. His research interests include image processing, computer vision, machine learning, and deep learning.

How to cite this article: J. Kim, K. Yun, H.-G. Choo, H. Mun, and K. Kong, *3D Gaussian rasterization for direct rendering of view-multiplexed images on multiview displays*, ETRI Journal **48** (2026), 711–722, DOI 10.4218/etrij.2025-0246

22337326, 2026, 4, Downloaded from https://onlinelibrary.wiley.com/doi/10.4218/etrij.2025-0246 by Electronics And Telecommunications, Wiley Online Library on [15/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License