3DGS^3: Joint Super Sampling and Frame Interpolation for Real-Time Large-Scale 3DGS Rendering

Yibo Zhao[1*], Fan Gao[1*], Youcheng Cai[1(✉)], Ligang Liu[1]

University of Science and Technology of China
{eabo,agno3}@mail.ustc.edu.cn, {caiyoucheng,lgliu}@ustc.edu.cn

Abstract. 3D Gaussian Splatting (3DGS) enables high-quality real-time 3D rendering but faces challenges in efficiently scaling to ultra-dense scenes and high resolutions due to computational bottlenecks that limit its use in latency-sensitive applications. Instead of optimizing the splatting pipeline itself, we propose **3DGS^3**, a unified post-rendering framework that jointly performs super sampling and frame interpolation through differentiable processing of low-resolution outputs to achieve both high-resolution and high-frame-rate rendering. Our **Gradient - Aware Super Sampling (GASS)** module leverages the continuous differentiability of 3DGS to extract image gradients that guide a GRU-based refinement network to enable high-fidelity super sampling. Furthermore, a **Lightweight Temporal Frame Interpolation (LTFI)** module based on a compact U-Net-like backbone fuses temporal and differentiable spatial cues from consecutive frames to synthesize temporally coherent intermediate frames. Experiments on public datasets demonstrate that 3DGS^3 achieves superior rendering efficiency and visual quality when compared with state-of-the-art methods and remains compatible with existing 3DGS acceleration techniques. The project code is available at: `https://github.com/Eabo20001/3DGS3`.

Keywords: Gaussian splatting · Super sampling · Frame interpolation

1 Introduction

3D Gaussian Splatting (3DGS) [29] has recently emerged as a powerful representation for real-time, high-quality 3D reconstruction by modeling scenes as a collection of Gaussian primitives. Although 3DGS enables real-time rendering across various domains, its performance is constrained by the high parameter count and inherent algorithmic inefficiencies, thereby limiting its broader adoption in latency-sensitive applications such as virtual reality (VR) [14, 25], augmented reality (AR) [36], and mobile devices [34].

*Equal contribution.
✉Corresponding author.

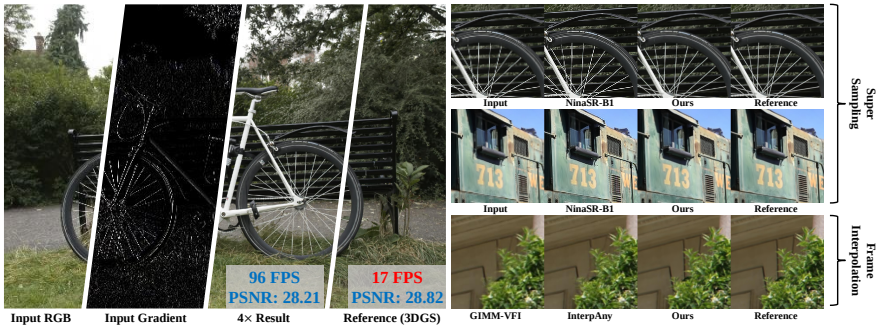


Fig. 1: We propose a novel post-rendering framework to accelerate 3DGS rendering. Leveraging spatio-temporal information, it jointly performs super sampling and frame interpolation over rendered outputs to achieve efficient acceleration while preserving high rendering quality.

Recent efforts to improve 3D Gaussian Splatting (3DGS) rendering efficiency can be broadly categorized into three major directions: (1) compact representations: methods such as GES [19] and DisC-GS [47] modify Gaussian primitives, through generalized exponential splats or boundary-aware formulations, to achieve more compact representations. (2) Gaussian reduction: methods such as LightGaussian [11] and Mini-Splatting [12] prune redundant splats based on spatial distributions to maintain visual quality with fewer primitives. (3) rendering acceleration: AdR-Gaussian [52] introduces adaptive-radius filtering, while GSCore [34] leverages hardware-level optimizations to accelerate rasterization. Despite these advances, a fundamental bottleneck persists: achieving real-time rendering remains challenging, particularly when scaling to ultra-dense scenes or ultra-high-resolution outputs (e.g., millions of splats or 4K rendering). These factors impose substantial computational overhead, preventing 3DGS from achieving the responsiveness required for interactive experiences.

Rather than further optimizing the splatting pipeline itself, we investigate ways to accelerate 3DGS from the perspective of post-rendering, drawing inspiration from modern real-time graphics pipelines. In conventional rendering, commercial systems such as DLSS [39], XeSS [24], and FSR [2] achieve high-quality real-time performance by employing super sampling and frame interpolation techniques to reconstruct 4K resolution or high frame rates from low-resolution inputs. This paradigm shift motivates us to explore whether a similar approach can be effectively applied to 3DGS. However, directly integrating such methods into 3DGS is challenging, as a naïve integration often degrades quality and cannot accommodate the inherently continuous nature of Gaussian representations.

To address these challenges, we propose 3DGS^3, a unified framework that jointly performs **S**uper **S**ampling and frame interpolation for real-time large-scale 3DG**S** rendering. Inspired by super sampling and frame interpolation techniques in conventional rendering pipelines, 3DGS^3 performs differentiable post-rendering on low-resolution 3DGS outputs, thereby achieving both high-resolution

and high-frame-rate rendering without modifying the underlying splatting pipeline (see Fig. 1). Specifically, we design a **Gradient-Aware Super Sampling (GASS)** that leverages the continuous differentiability of Gaussian splats to extract image gradients serving as auxiliary cues. These gradient features guide image interpolation and a GRU-based refinement network to efficiently reconstruct high-resolution renderings while preserving perceptual fidelity. In addition, a **Lightweight Temporal Frame Interpolation (LTFI)** based on a compact U-Net-like backbone fuses temporal and differentiable spatial cues from consecutive frames to synthesize intermediate images, thereby substantially enhancing rendering frame rates. Extensive experiments on real-world datasets demonstrate that 3DGS^3 achieves superior rendering efficiency and visual fidelity when compared with existing 3DGS acceleration and upscaling approaches.

Our main contributions are summarized as follows:

- **A new paradigm for 3DGS acceleration.** To the best of our knowledge, this work presents the first post-rendering framework that jointly performs super sampling and frame interpolation for 3DGS rendering.
- **Gradient-aware super sampling.** We exploit the differentiable properties of Gaussians to extract continuous image gradients, which guide image interpolation and refinement to achieve efficient and high-quality rendering.
- **Lightweight temporal frame interpolation.** We propose a lightweight U-Net-like interpolation module that fuses temporal and differentiable spatial cues to generate high-quality intermediate frames.

2 Related Work

Accelerating 3DGS Rendering. 3D Gaussian Splatting (3DGS) [29] offers a differentiable and explicit representation that enables real-time, high-quality rendering and has been applied to autonomous driving, human avatars, and dynamic scenes. In this section, we review efforts to accelerate 3DGS rendering, which primarily focus on three aspects: (1) compact representations, (2) Gaussian reduction, and (3) rendering acceleration.

Compact representations. These methods design more expressive and compact primitives to reduce redundancy. GES [19] introduces generalized exponential splatting, while DisC-GS [47] models depth and color discontinuities using a discontinuity-aware Gaussian for more realistic rendering. BG-Triangle [53] vectorizes Gaussians with Bézier triangles, and DRK [23] adopts deformable radial kernels to improve modeling flexibility. Scaffold-GS [41] uses anchor-based adaptive parameters, and LocoGS [50] applies locality-aware compression for faster rendering.

Gaussian reduction. These methods prune redundant primitives to reduce computational cost. EAGLES [15] combines quantized embeddings and pruning for a ten- to twenty-fold reduction. LightGaussian [11] compresses unbounded Gaussians for over 200 FPS rendering, while Mini-Splatting [12] and PUP-3DGS [21] perform budgeted and uncertainty-based pruning. RadSplat [45] uses radiance-aware selection, achieving 900 FPS rendering.

Rendering acceleration. Other methods enhance pipeline efficiency through hardware and rasterization optimizations. ADR-Gaussian [52] adapts Gaussian radius to reduce overdraw, Speedy-Splat [20] exploits sparsity of pixels and primitives for fast splatting, FlashGS [13] optimizes GPU pipelines for large-scale scenes, and GSCore [34] introduces hardware-level optimizations.

Despite these advances, real-time rendering under ultra-dense or ultra-high-resolution settings remains challenging. UpscaleGS [44] integrates super sampling into 3DGS but requires scene-specific retraining. In contrast, our modular, scene-agnostic post-rendering framework performs differentiable processing on low-resolution 3DGS outputs to generate high-resolution and high-frame-rate results, remaining complementary to existing accelerations.

Real-time Super Sampling. Traditional anti-aliasing techniques such as MSAA [1], MLAA [48], and SMAA [26] enhance image quality through spatial averaging or edge-aware filtering, while temporal super sampling [58] reuses samples across frames. However, these heuristic methods have difficulty preserving fine details. Neural approaches like DLSS [39] and XeSS [24] leverage temporal and spatial cues for reconstruction, along with NSRR [57], NSRD [35], and FuseSR [64], which enhance temporal stability through end-to-end learning. Recently, RDG [62] adopts decoupled G-buffer guidance to improve temporal coherence and overall rendering consistency.

Real-time Frame Generation. Early frame synthesis relied on temporal reprojection and 3D warping [3, 42, 49], reusing past frames to increase frame rates but suffering from ghosting and occlusion issues. Neural methods [6, 7, 17, 37, 55, 56] instead learn motion-aware features to predict or interpolate frames, achieving higher temporal consistency, although they depend on rasterized G-buffers, limiting applicability to 3DGS. Complementary progress in video frame interpolation, ranging from phase- or kernel-based [10, 33, 43, 46] to optical flow–based approaches [4, 18, 28, 40], further motivates our differentiable post-rendering design for temporal upsampling.

3 Preliminaries

3D Gaussian Splatting. 3DGS [29] represents a 3D scene as a comprehensive set of anisotropic Gaussian primitives. Each Gaussian primitive is defined as:

$$G(\mathbf{x}) = \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}_c)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{x}_c)\right], \tag{1}$$

where $\mathbf{x}_c$ represents the Gaussian center and $\mathbf{\Sigma}$ is the covariance matrix, which can be factorized as $\mathbf{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$ for a rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$. Subsequently, the Gaussians are projected into the image space via EWA Splatting [65], which enables efficient rasterization-based rendering. The resulting 2D Gaussians are depth-sorted, and pixel colors are computed using α-blending:

$$I = \sum_{i\in N} c_i\alpha_i T_i = \sum_{i\in N} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \tag{2}$$

where α_i is computed by the Gaussian multiplied by the opacity and c_i is the view-dependent color predicted from spherical harmonics (SH) coefficients.

3DGS Image Gradients. UpscaleGS [44] proposes utilizing the analytical image-space gradients of Gaussians, obtained from the exact derivatives of the smooth, continuous signal. Assuming N Gaussians represent the scene, the rendered image $I(x, y)$ is differentiable with respect to the 2D spatial coordinates (x, y):

$$\begin{aligned} \frac{\partial I(x,y)}{\partial x} &= \sum_{i=1}^{N} c_i \left(\frac{\partial T_i}{\partial x}\alpha_i + T_i \frac{\partial \alpha_i}{\partial x}\right), \\ \frac{\partial I(x,y)}{\partial y} &= \sum_{i=1}^{N} c_i \left(\frac{\partial T_i}{\partial y}\alpha_i + T_i \frac{\partial \alpha_i}{\partial y}\right). \end{aligned} \tag{3}$$

Here, the differential of T_i can be recursively expanded as:

$$\frac{\partial T_i}{\partial x} = -\frac{\partial \alpha_{i-1}}{\partial x} T_{i-1} + (1 - \alpha_{i-1}) \frac{\partial T_{i-1}}{\partial x}, \tag{4}$$

and the computation of $\frac{\partial T_i}{\partial y}$ follows an analogous form. The image-space gradients are obtained by blending the weighted per-Gaussian gradients.

4 Method

In this section, we introduce **3DGS^3**, a unified framework that jointly performs super sampling and frame interpolation to accelerate 3DGS rendering. The overall structure of our pipeline is illustrated in Figure 2. First, we present a **Gradient-Aware Super Sampling (GASS)** module that leverages the differentiable nature of 3DGS-rendered images to achieve high-quality super sampling. Furthermore, we propose a **Lightweight Temporal Frame Interpolation (LTFI)** module based on a compact U-Net-like backbone to reduce computational cost by effectively leveraging temporal and differentiable spatial cues. The two modules work collaboratively to reduce the average rendering time per frame, thereby improving overall rendering efficiency.

4.1 Gradient-Aware Super Sampling

Unlike UpscaleGS [44], which employs image-space gradients to determine a bicubic spline function fitted to neighboring pixels around the target pixel for interpolation, we propose a GASS network, consisting of two main components: (1) Gradient-aware interpolation and (2) Geometry-temporal recurrent refinement, that more effectively and robustly leverages geometric and historical priors to achieve superior super sampling performance.

Gradient-Aware Interpolation. Inspired by Hermite interpolation [30], we propose a novel gradient-aware interpolation method. Hermite interpolation incorporates both function values and their corresponding derivatives at the interpolation nodes. For each interpolation node, the polynomial is constrained

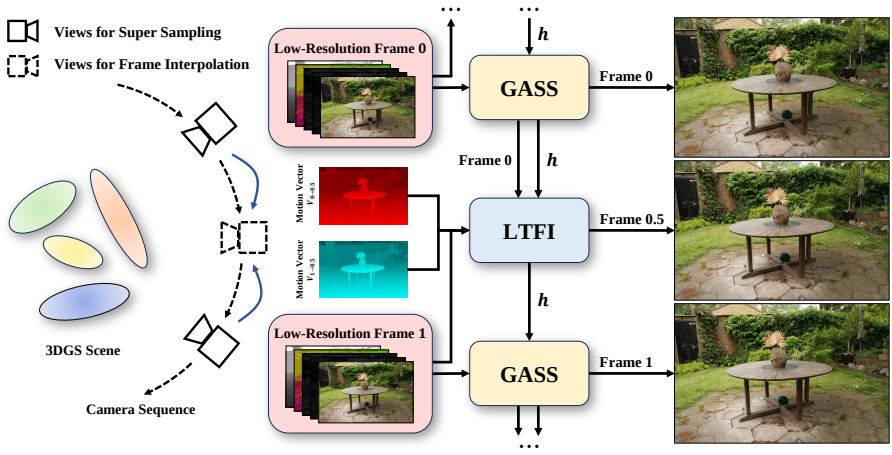


Fig. 2: Overview of our approach. (1) The **Gradient-Aware Super Sampling (GASS)** module processes the low-resolution frames rendered by 3DGS by combining them with image gradients and the historical feature to generate high-resolution images. (2) The **Lightweight Temporal Frame Interpolation (LTFI)** module uses the high-resolution image together with the subsequent low-resolution frame to interpolate intermediate frames.

to match the function values and its derivatives up to a specified order of the original function.

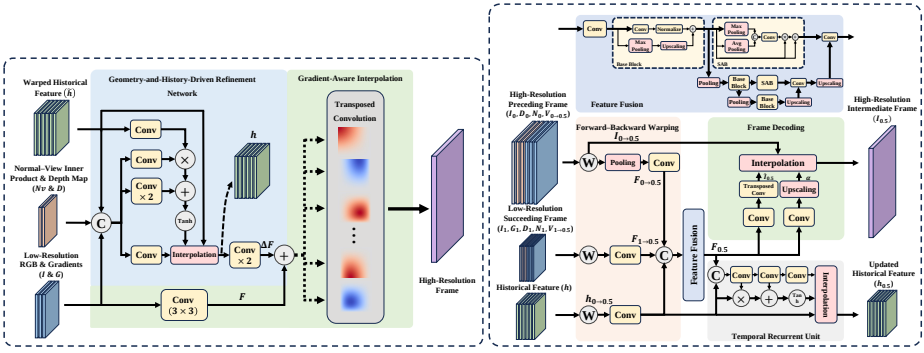
For a given pixel (x_0, y_0), we consider it along with its eight adjacent pixels, forming a 3×3 block of interpolation nodes. Although two-dimensional interpolation schemes may vary depending on node distribution and derivative availability, we consider a bivariate polynomial $f(x_0, y_0)$ that satisfies the following conditions:

$$\begin{aligned} f(x_0+i, y_0+j) &= I(x_0+i, y_0+j), \\ \frac{\partial}{\partial x} f(x_0+i, y_0+j) &= \frac{\partial}{\partial x} I(x_0+i, y_0+j), \\ \frac{\partial}{\partial y} f(x_0+i, y_0+j) &= \frac{\partial}{\partial y} I(x_0+i, y_0+j), \end{aligned} \tag{5}$$

where $i, j \in \{-1, 0, 1\}$. Thus, the interpolated image $\hat{I}$ can be viewed as evaluating different local polynomials $f(x, y)$ at their corresponding spatial positions. For simplicity, we define that under an $S\times$ super sampling setting, the 3×3 block centered at (x_0, y_0) in I corresponds to an $S \times S$ block centered at $(\hat{x}_0, \hat{y}_0)$ in $\hat{I}$:

$$I_{3\times3}(x_0, y_0) \leftrightarrow \hat{I}_{3\times3}(\hat{x}_0, \hat{y}_0). \tag{6}$$

Consequently, each pixel within the block can be expressed as a linear combination of the low-resolution intensities and their spatial gradients, weighted by a



(a) The architecture of the GASS module. (b) The architecture of the LTFI module.

Fig. 3: The detailed architecture of the (a) GASS module, which consists of a gradient-aware interpolation module and a geometry-temporal recurrent refinement network; and (b) LTFI module, which consists of a Forward–Backward Warping block, a Feature Fusion Network, and a Temporal Recurrent Unit.

coefficient matrix $\mathbf{M}$:

$$\mathrm{Vec}(\hat{I}_{S\times S}(\hat{x}_0,\hat{y}_0)) = \mathbf{M}\begin{bmatrix}\mathrm{Vec}(I_{3\times3}(x_0,y_0))\\ \mathrm{Vec}(\frac{\partial}{\partial x}I_{3\times3}(x_0,y_0))\\ \mathrm{Vec}(\frac{\partial}{\partial y}I_{3\times3}(x_0,y_0))\end{bmatrix} \tag{7}$$

where $\mathrm{Vec}(\cdot)$ denotes the column-wise vectorization operator. This operation can be implemented by cascading a convolution layer with a transposed-convolution layer:

$$\hat{I} = \mathrm{TConv}(F), F = \mathrm{Conv}(I), \tag{8}$$

where $\mathrm{Conv}(\cdot)$ denotes a 3×3 convolution layer and $\mathrm{TConv}(\cdot)$ denotes an $S\times S$ transposed-convolution layer. We interpret F as the coefficient vector in the polynomial space, while $\mathrm{TConv}(\cdot)$ serves as a linear functional from the dual space, mapping elements of the polynomial space to pixel-wise intensities. Notably, the number of channels in F is chosen to exceed its theoretical minimum dimension $\min\{S^2, 27\}$, ensuring that $\mathbf{M}$ in Eq. (7) possesses at least full column or row rank. This design choice enables the network to capture higher-order relationships.

Geometry-Temporal Recurrent Refinement. While the introduction of gradients enhances the recovery of fine image details, these gradients are themselves derived from the partial derivatives of the original continuous image function. Consequently, they remain constrained by the sampling frequency, making it difficult to avoid aliasing artifacts. Recent works [56, 64] have demonstrated that incorporating G-buffers and temporal cues can significantly improve reconstruction quality. Motivated by this, we propose a Geometry-Temporal Recurrent Refinement Network to further enhance the interpolated outputs.

As illustrated in Fig. 3 (a), the proposed network employs a modified gated recurrent unit (GRU) [9] architecture. The network takes as input the current

low-resolution frame I and its gradients G, the depth map D, the normal–view inner product map Nv, and the warped historical feature $\tilde{h}$ from the previous frame, which is aligned to the current frame. The refinement process yields an updated historical feature representation for the current frame:

$$h = \text{RefineNet}(I, G, D, Nv, \tilde{h}). \tag{9}$$

Finally, the current historical feature h is utilized to generate a residual feature ΔF via a convolutional operation:

$$\Delta F = \text{Conv}(h), \tag{10}$$

and the updated feature is then obtained as $F \leftarrow F + \Delta F$, which follows the formulation in Eq. (8).

As mentioned above, F represents the coefficient vector in the polynomial space. The refinement network updates F by leveraging both current-frame features and temporally propagated information, resulting in more robust and temporally consistent outputs. This design provides two main advantages: (1) it remains lightweight, ensuring computational efficiency, and (2) it effectively leverages the geometric and temporal information available in the current frame, thereby enhancing overall interpolation accuracy.

Loss Function Design. To ensure both accurate super sampling and faithful detail reconstruction, the loss function for GASS consists of two components: the Charbonnier loss [8] $\mathcal{L}_C$ provides the primary photometric constraint, and the Laplace loss $\mathcal{L}_{lap}(X, Y) = \left\|\nabla^2 X - \nabla^2 Y\right\|_1$ emphasizes fine-grained detail consistency. The overall loss function for GASS is formulated as:

$$\mathcal{L} = \mathcal{L}_C + \lambda_1 \mathcal{L}_{lap}, \tag{11}$$

where $\lambda_1 = 0.1$ is the weighting factor.

4.2 Lightweight Temporal Frame Interpolation

To further enhance rendering efficiency, we introduce a LTFI module that leverages the inherent spatio-temporal coherence of 3DGS. Unlike existing video frame interpolation methods [18, 28, 32] that employ large and computationally expensive networks, our LTFI is explicitly designed to be **lightweight and efficient**. It effectively exploits gradient and geometric priors, along with historical temporal information, to achieve real-time interpolation. Given two frames I_0 and I_1, the objective is to predict the intermediate frame $I_{0.5}$. The network comprises three key components: (1) forward–backward warping, (2) a feature fusion network, and (3) a temporal recurrent unit.

Forward–Backward Warping. We first warp the preceding high-resolution frame and the subsequent low-resolution frame to the intermediate time step ($t = 0.5$). Specifically, the high-resolution image I_0, depth map D_0, and normal map N_0 from frame 0 are warped to the intermediate frame, and the low-resolution image I_1, depth map D_1, normal map N_1, and gradient map G_1 from frame 1

are projected in the same manner. The warping operation is guided by motion vectors derived from camera poses. For a pixel x in frame 1, the motion vector is defined as:

$$V_{1\to 0.5}[x] = \text{Proj}(\text{UnProj}(x, P_1), P_{0.5}) - x, \tag{12}$$

where P_i denotes the camera parameters of frames i, respectively. The warped gradient map is computed as:

$$G_{1\to 0.5} = \nabla V_{1\to 0.5} \cdot w(G_1, V_{1\to 0.5}), \tag{13}$$

where $w(\cdot)$ denotes the warping operator and ∇ represents spatial differentiation. We employ high-resolution features from the preceding frame and low-resolution features from the subsequent frame to minimize latency and facilitate parallel processing, thereby preserving temporal consistency and improving computational efficiency.

Feature Fusion Network. The warped inputs from both directions are encoded by convolutional layers into $F_{0\to 0.5}$ and $F_{1\to 0.5}$, which are then passed to a Feature Fusion Network (FFN)—a lightweight U-Net-like architecture illustrated in Fig. 3 (b). Each base block integrates max-pooling and upscaling operations to extract low-frequency structures. To maintain efficiency, we incorporate a Spatial Attention Block (SAB) [54] into the skip connections to enhance feature discriminability under complex occlusions and suppress irrelevant regions. For upscaling, we employ the pixel shuffle operation to increase spatial resolution via channel rearrangement, which is computationally cheaper and avoids artifacts compared to transposed-convolutions.

The FFN outputs a blending weight map α and a fused color prediction $\hat{I}_{0.5}$. The final interpolated high-resolution frame is computed as:

$$I_{0.5} = \alpha \hat{I}_{0.5} + (1 - \alpha) I_{0\to 0.5}. \tag{14}$$

This adaptive blending formulation balances the reliability of warped content and the accuracy of fused predictions, preserving fine details while mitigating temporal flicker.

Temporal Recurrent Unit. To further enforce temporal consistency, we introduce a Temporal Recurrent Unit (TRU) that explicitly models inter-frame dependencies. The TRU is a modified GRU, which takes as input the warped historical hidden state $h_{0\to 0.5}$ and the current fused feature $F_{0.5}$, producing an updated hidden representation:

$$h_{0.5} = U(h_{0\to 0.5}, F_{0.5}), \tag{15}$$

where $U(\cdot)$ denotes the recurrent update function. The hidden state $h_{0.5}$ progressively accumulates image features and geometric priors from previous frames, forming a compact spatio-temporal representation of the scene. This design facilitates consistent motion estimation and appearance interpolation across frames, ensuring smooth and stable rendering in 3DGS scenes.

Loss Function Design. To improve image detail restoration and boost temporal consistency across image sequences, we design the following loss function. A

Charbonnier loss $\mathcal{L}_C$ [8] is employed to improve the robustness of reconstructed images against outliers. A perceptual loss $\mathcal{L}_{\text{VGG}}$ [27] is used to enhance the perceptual quality of reconstructed images through high-level feature constraints. To constrain the adaptive blending process, ensuring that it restores only unreliable regions in $I_{0\to0.5}$, we incorporate an $\mathcal{L}_2$ regularization term on α, defined as $\mathcal{L}_\alpha = \|\alpha\|_2$. To further mitigate inter-frame flickering, an occlusion-aware loss is additionally introduced, formulated as follows:

$$\mathcal{L}_{\text{OCC}} = \sum_{j=0,1} \mathcal{L}_1(w(I_{0.5}, V_{0.5\to j}), I_j). \tag{16}$$

Finally, the overall loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_C + \lambda_1 \mathcal{L}_{\text{VGG}} + \lambda_2 \mathcal{L}_{\text{OCC}} + \lambda_3 \mathcal{L}_\alpha, \tag{17}$$

where $\lambda_1 = \lambda_3 = 0.2$ and $\lambda_2 = 0.05$ are weighting factors.

5 Experiment

5.1 Implementation Details

Our method is implemented based on the 3DGS framework [29] and employs RaDe-GS [60] for rendering depth and normal maps. Gradient computation, motion vector estimation, and warping operations are implemented through custom CUDA kernels. Our model is trained with the Adam optimizer at a learning rate of 10^{-3}. For inference, the network is deployed with TensorRT for accelerated execution, where super sampling and frame interpolation are parallelized to achieve optimal efficiency. All experiments are conducted on an NVIDIA RTX 3090 GPU.

5.2 Datasets and Metrics

We evaluate our method on three widely used public benchmarks: Mip-NeRF 360 [5], Tanks and Temples [31], and Deep Blending [22]. These datasets cover diverse scenarios, ranging from unconstrained outdoor environments to complex indoor scenes, thereby offering substantial diversity. Our model is trained on the *Garden* scene from Mip-NeRF 360 and evaluated on all available scenes, with average performance reported over 300-frame sequences.

We use standard PSNR and SSIM as quantitative metrics. To assess runtime efficiency, we also report the average FPS. Since our method operates as a post-rendering enhancement, the ground-truth images are generated using standard 3DGS [29] renderings. This setup ensures that all quality differences stem exclusively from the super sampling or frame interpolation algorithms, rather than discrepancies in the underlying 3D reconstruction, thereby facilitating a fair comparison with other super sampling and frame interpolation methods. To further validate the effectiveness of our approach under real-world conditions, we also include an evaluation using real-world captured images as ground truth.

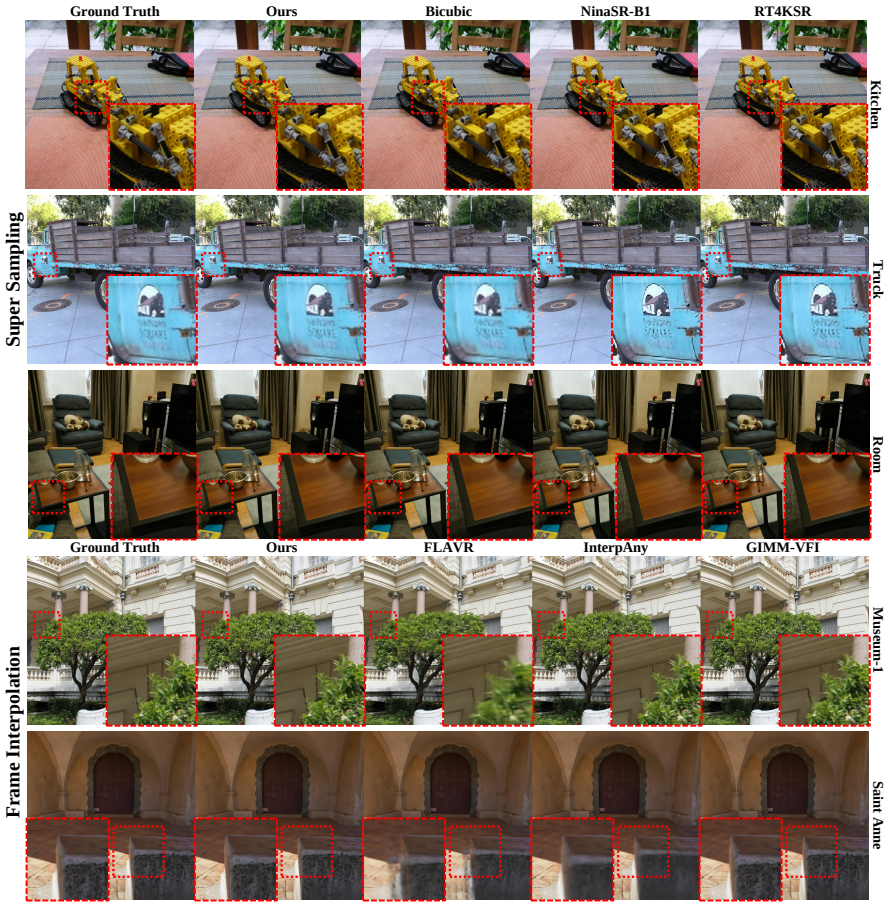


Fig. 4: Qualitative comparison of super sampling and frame interpolation results. Our method achieves superior detail preservation in super sampling and smoother temporal consistency in frame interpolation compared to existing approaches.

To thoroughly evaluate the effectiveness of our approach, we compare against the following categories of methods: (1) **Traditional Interpolation**: Bicubic interpolation; (2) **Learning-based Super Sampling**: NinaSR-B1 [16], RT4KSR [59], ECBSR [61], and UpscaleGS [44]; (3) **Video Frame Interpolation**: FLAVR [28], InterpAny [63], and GIMM-VFI [18]; and (4) **3DGS Acceleration**: Mini-Splatting [12], Speedy-Splat [20], and AdR-Gaussian [52].

5.3 Comparisons

Table 1 presents a quantitative comparison of our method against various state-of-the-art approaches across three benchmark datasets. As shown, our method achieves the highest FPS among all super sampling and frame interpolation

Table 1: Quantitative comparisons on the Mip-NeRF 360 [5], Deep Blending [22], and Tanks and Temples [31] datasets. All super sampling methods employ a 4× scale. Our method achieves the highest FPS among all approaches while maintaining highly competitive visual quality.

Method	Mip-NeRF 360 4096 × 2160			Deep Blending 2560 × 1440			Tanks and Temples 2560 × 1440		
	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑
3DGS (HR)	-	-	17.35	-	-	41.57	-	-	50.40
Ours	**49.22**	**0.995**	**96.47**	40.85	0.983	**226.50**	37.64	0.978	**246.35**
Bicubic	47.63	0.992	84.20	36.41	0.942	149.26	35.73	0.954	165.59
UpscaleGS	41.45	0.980	77.50	35.89	0.947	129.34	33.91	0.933	138.89
NinaSR-B1	40.49	0.985	4.47	30.41	0.901	16.60	29.88	0.923	16.77
RT4KSR	43.82	0.986	12.00	33.85	0.922	39.47	32.73	0.936	40.14
ECBSR	44.97	0.989	47.30	35.83	0.940	66.73	34.73	0.951	63.28
FLAVR	N/A	N/A	N/A	34.73	0.939	0.67	31.14	0.939	0.67
InterpAny	36.41	0.962	1.25	**48.43**	**0.996**	4.93	**40.11**	**0.992**	4.93
GIMM-VFI	42.57	0.989	1.59	42.34	0.987	4.83	38.47	0.985	4.76

Table 2: Quality comparison of 3DGS methods integrated with our approach. For scenes where the real-world captured images ground truth images have a maximum resolution lower than 4096 × 2160, we upsample them to this target resolution via bicubic interpolation followed by cropping to ensure consistent evaluation.

Method	2560 × 1440						4096 × 2160					
	w/o Ours			w/ Ours			w/o Ours			w/ Ours		
	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑
3DGS	28.95	0.859	47.11	28.92	0.858	**188.43**	30.68	0.906	17.35	30.47	0.899	**96.47**
Mini-Splatting	28.88	0.878	181.83	28.76	0.875	**463.90**	29.88	0.895	56.00	29.66	0.891	**158.78**
Speedy-Splat	28.33	0.858	146.81	28.20	0.857	**261.81**	28.33	0.858	79.93	28.20	0.857	**130.68**
AdR-Gaussian	29.68	0.886	127.10	29.67	0.885	**260.60**	28.38	0.872	58.27	28.07	0.867	**128.70**

Table 3: Standalone runtime of each method.

Method	4096 × 2160	2560 × 1440	Method	4096 × 2160	2560 × 1440
3DGS HR Rendering	57.64 ms	24.06 ms	NinaSR-B1	213.18 ms	53.93 ms
3DGS LR Rendering	10.62 ms	6.30 ms	RT4KSR	72.97 ms	19.03 ms
Bicubic	1.51 ms	0.40 ms	ECBSR	10.78 ms	8.68 ms
UpscaleGS	2.28 ms	1.67 ms	FLAVR	N/A	744.08 ms
GASS (ours)	5.78 ms	1.45 ms	InterpAny	394.35 ms	98.48 ms
LTFI (ours)	11.25 ms	2.81 ms	GIMM-VFI	310.64 ms	100.64 ms

Table 4: Cross-dataset generalization with different training-scene choices.

Method	Mip-NeRF 360 4096 × 2160		Deep Blending 2560 × 1440		Tanks and Temples 2560 × 1440	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Train on Garden scene	49.22	0.995	41.03	0.983	37.64	0.978
Train on Bedroom scene	49.31	0.997	40.94	0.979	37.69	0.976
Train on Mip-NeRF 360 dataset	—	—	41.11	0.988	37.65	0.975

methods, demonstrating its superior runtime efficiency. Table 3 reports the standalone runtime of each key step. Moreover, our approach achieves higher image quality than existing learning-based super sampling methods.

We further compare our method with UpscaleGS [44]. UpscaleGS inherently couples super sampling with the 3DGS reconstruction process to achieve satisfactory performance, whereas our method is a plug-and-play post-processing

module. For a fair comparison, we evaluate UpscaleGS without per-scene training by independently reproducing its core algorithm, since its code has not yet been publicly released. As shown in Table 1, UpscaleGS yields lower quality than our method across all evaluated datasets. Benefiting from frame interpolation and our parallel pipeline design, our method also achieves a higher frame rate at 4K resolution.

By default, our models are trained only on 300 contiguous frames from a loop-shaped trajectory from the Garden scene. Owing to their small parameter sizes and local-feature focus, they can be trained on a single scene and still generalize effectively. Table 4 reports additional training-scene choices, with Garden from Mip-NeRF 360 and Bedroom from Deep Blending excluded from the corresponding test data. The similar PSNR and SSIM values further support our cross-scene generalization capability.

Our method surpasses all interpolation approaches in terms of FPS by a large margin, while achieving highly competitive visual quality. It is worth noting that InterpAny [63] and other interpolation methods attain high quality by employing large networks and leveraging high-resolution ground-truth frames as inputs. In contrast, our lightweight approach processes low-resolution inputs and synthesizes the intermediate frame from a super sampled previous frame and a low-resolution subsequent frame, without access to ground-truth supervision. Despite this challenging setting, our method produces results with fidelity comparable to these heavyweight techniques. As illustrated in Fig. 4, the qualitative comparisons confirm that our method produces superior details and smoother temporal transitions in both super sampling and frame interpolation tasks, underscoring the effectiveness of our design.

To further validate the efficacy of our method, we conduct an additional evaluation using real-world captured images as ground truth. As shown in Table 2, our method incurs only negligible degradation in PSNR and SSIM compared to the original 3DGS renderings, even under 4× super sampling and frame interpolation. These results confirm that our method maintains high visual fidelity while delivering significant rendering acceleration, demonstrating its applicability to real-world deployment.

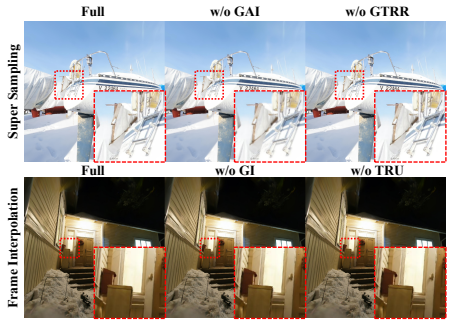
5.4 Integration with 3DGS Acceleration Methods

To further validate the generality and scalability of our approach, we integrate it with several representative 3DGS acceleration methods. Table 2 summarizes the rendering efficiency, measured in FPS, after incorporating our post-rendering module into these acceleration frameworks. The experimental results show that our method consistently improves the rendering efficiency of all evaluated techniques, confirming its strong synergy with diverse acceleration strategies. This integration offers a flexible and robust solution for achieving high-frame-rate 3DGS rendering.

Table 5: Ablation study on key components of our approach.

	PSNR↑	SSIM↑	FPS↑
FULL	**40.85**	**0.983**	226.50
w/o GASS	40.34	0.984	75.23
w/o LTFI	42.13	0.985	158.66
w/o GAI	36.81	0.960	226.50
w/o GTRR	39.39	0.976	226.50
w/o GI	39.35	0.976	**233.85**
w/o TRU	39.63	0.977	227.21

Fig. 5: Ablation of GAI, GTRR, GI, and TRU components.



5.5 Ablation Studies

To evaluate the effectiveness of each key component in our approach, we conduct ablation experiments on the Deep Blending [22] dataset at a resolution of 2560×1440. We remove the following components and compare them against the full model: (1) without Gradient-Aware Super Sampling (w/o GASS), (2) without Lightweight Temporal Frame Interpolation (w/o LTFI), (3) without Gradient-Aware Interpolation in GASS (w/o GAI), (4) without Geometry-Temporal Recurrent Refinement in GASS (w/o GTRR), (5) without Gradient Information in LTFI (w/o GI), and (6) without Temporal Recurrent Unit in LTFI (w/o TRU).

The quantitative results in Table 5 indicate that neither the super sampling nor the frame interpolation branch alone can achieve optimal performance. For the super sampling branch, removing GASS or GAI leads to a significant degradation in PSNR and SSIM. For frame interpolation, eliminating GI or TRU slightly improves the frame rate but leads to a noticeable decline in image quality. As illustrated in Fig. 5, the absence of GASS, GAI, GI, or TRU causes blurred textures and visible temporal artifacts, further highlighting the importance of these components.

5.6 Extending to Large-Scale Datasets

To further evaluate the scalability of our approach, we conduct experiments on larger scenes from UrbanScene3D [38] and Mill 19 [51], including the Residence, Sci-Art, Building, and Rubble scenes. These scenes contain more than 10M Gaussian primitives, making high-resolution rendering substantially more computationally demanding than the standard benchmark scenes.

As shown in Table 6, 3DGS struggles to maintain real-time rendering speed on these large-scale scenes, especially at 4K resolution. In contrast, our method consistently improves the rendering efficiency across all evaluated scenes and enables real-time performance at both 2K and 4K resolutions. Meanwhile, the

Table 6: Rendering efficiency on large-scale scenes from UrbanScene3D [38] and Mill 19 [51]. Each scene contains over 10M Gaussian primitives. Our method maintains real-time rendering at both 2K and 4K resolutions while preserving comparable visual quality.

Scene	#GS	2560 × 1440						4096 × 2160					
		w/o Ours			w/ Ours			w/o Ours			w/ Ours		
		PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑	PSNR↑	SSIM↑	FPS↑
Sci-Art	10.8M	20.72	0.775	24.59	20.70	0.773	**83.83**	20.69	0.787	9.89	20.69	0.786	**61.33**
Residence	10.1M	21.36	0.739	24.19	21.22	0.718	**82.47**	21.76	0.701	9.58	21.70	0.696	**60.00**
Rubble	10.8M	22.16	0.770	34.13	22.13	0.768	**87.78**	22.30	0.746	11.72	22.28	0.743	**65.83**
Building	10.7M	20.47	0.717	24.33	20.39	0.700	**81.42**	20.43	0.738	10.20	20.42	0.736	**59.56**

PSNR and SSIM remain close to the original high-resolution renderings, indicating that the efficiency gain does not come at the cost of severe quality degradation. These results demonstrate that 3DGS^3 can effectively scale to high-Gaussian-count scenes and support real-time large-scale 3DGS rendering.

6 Conclusion

In this paper, we present 3DGS^3, a novel framework that reexamines 3DGS acceleration from the perspective of post-rendering. Unlike prior methods that focus on optimizing the splatting pipeline itself, our approach introduces a unified framework for super sampling and frame interpolation, enabling the efficient reconstruction of high-resolution and high-frame-rate renderings from low-resolution 3DGS outputs. Specifically, we propose a gradient-aware super sampling module that leverages the spatial gradients of Gaussian splats for high-fidelity detail reconstruction. In addition, a lightweight temporal frame interpolation module based on a compact U-Net-like architecture synthesizes intermediate frames by fusing temporal and differentiable spatial cues from consecutive inputs. Extensive experiments demonstrate that our method outperforms state-of-the-art approaches in both rendering quality and efficiency, while remaining fully compatible with existing 3DGS acceleration techniques, thereby advancing real-time rendering performance with reduced computational overhead. We note that our current method focuses primarily on static scenes; extending it to dynamic 3DGS representations remains a promising direction for future work. Additionally, we aim to extend our framework to mobile platforms by developing compact and lightweight model variants for efficient on-device deployment.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (U25A20444,62025207) and the Fundamental Research Funds for the Central Universities China (WK0010250100).

References

1. Akeley, K.: Reality engine graphics. In: Annual Conference on Computer Graphics and Interactive Techniques. pp. 109–116 (1993)
2. AMD: Amd fidelityfx super resolution. https://www.amd.com/en/technologies/fidelityfx-super-resolution (2021)
3. AMD: Amd fidelityfx super resolution 2.1. https://github.com/GPUOpen-Effects/FidelityFX-FSR2 (2022)
4. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3703–3712 (2019)
5. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
6. Briedis, K.M., Djelouah, A., Meyer, M., McGonigal, I., Gross, M., Schroers, C.: Neural frame interpolation for rendered content. ACM Transactions on Graphics **40**(6), 1–13 (2021)
7. Briedis, K.M., Djelouah, A., Ortiz, R., Meyer, M., Gross, M., Schroers, C.: Kernel-based frame interpolation for spatio-temporally adaptive rendering. In: SIGGRAPH Conference Papers. pp. 1–11 (2023)
8. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. International Journal of Computer Vision **61**(3), 211–231 (2005)
9. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
10. Ding, T., Liang, L., Zhu, Z., Zharkov, I.: Cdfi: Compression-driven network design for frame interpolation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8001–8011 (2021)
11. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z., et al.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. Advances in Neural Information Processing Systems **37**, 140138–140158 (2024)
12. Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians. In: European Conference on Computer Vision. pp. 165–181 (2024)
13. Feng, G., Chen, S., Fu, R., Liao, Z., Wang, Y., Liu, T., Hu, B., Xu, L., Pei, Z., Li, H., et al.: Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 26652–26662 (2025)
14. Franke, L., Fink, L., Stamminger, M.: Vr-splatting: Foveated radiance field rendering via 3d gaussian splatting and neural points. ACM on Computer Graphics and Interactive Techniques **8**(1), 1–21 (2025)
15. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In: European Conference on Computer Vision. pp. 54–71 (2024)
16. Gouvine, G.: NinaSR: Efficient small and large convnets for super-resolution. https://github.com/Coloquinte/torchSR/blob/main/doc/NinaSR.md (2021). https://doi.org/10.5281/zenodo.4868308
17. Guo, J., Fu, X., Lin, L., Ma, H., Guo, Y., Liu, S., Yan, L.Q.: Extranet: Real-time extrapolated rendering for low-latency temporal supersampling. ACM Transactions on Graphics **40**(6), 1–16 (2021)

18. Guo, Z., Li, W., Loy, C.C.: Generalizable implicit motion modeling for video frame interpolation. Advances in Neural Information Processing Systems **37**, 63747–63770 (2024)
19. Hamdi, A., Melas-Kyriazi, L., Mai, J., Qian, G., Liu, R., Vondrick, C., Ghanem, B., Vedaldi, A.: Ges: Generalized exponential splatting for efficient radiance field rendering. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19812–19822 (2024)
20. Hanson, A., Tu, A., Lin, G., Singla, V., Zwicker, M., Goldstein, T.: Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21537–21546 (2025)
21. Hanson, A., Tu, A., Singla, V., Jayawardhana, M., Zwicker, M., Goldstein, T.: Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5949–5958 (2025)
22. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics **37**(6), 1–15 (2018)
23. Huang, Y.H., Lin, M.X., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Deformable radial kernel splatting. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21513–21523 (2025)
24. Intel: Intel Arc Xe Super Sampling. `https://www.intel.cn/content/www/cn/zh/developer/topic-technology/gamedev/xess2.html` (2023)
25. Jiang, Y., Yu, C., Xie, T., Li, X., Feng, Y., Wang, H., Li, M., Lau, H., Gao, F., Yang, Y., et al.: Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In: SIGGRAPH Conference Papers. pp. 1–1 (2024)
26. Jimenez, J., Echevarria, J.I., Sousa, T., Gutierrez, D.: Smaa: Enhanced subpixel morphological antialiasing. In: Computer Graphics Forum. pp. 355–364 (2012)
27. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. pp. 694–711 (2016)
28. Kalluri, T., Pathak, D., Chandraker, M., Tran, D.: Flavr: Flow-agnostic video representations for fast frame interpolation. In: Winter Conference on Applications of Computer Vision. pp. 2071–2082 (2023)
29. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 139–1 (2023)
30. Kincaid, D.R., Cheney, E.W.: Numerical Analysis: Mathematics of Scientific Computing, vol. 2. American Mathematical Society (2009)
31. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
32. Kong, L., Jiang, B., Luo, D., Chu, W., Huang, X., Tai, Y., Wang, C., Yang, J.: Ifrnet: Intermediate feature refine network for efficient frame interpolation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1969–1978 (2022)
33. Lee, H., Kim, T., Chung, T.y., Pak, D., Ban, Y., Lee, S.: Adacof: Adaptive collaboration of flows for video frame interpolation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5316–5325 (2020)
34. Lee, J., Lee, S., Lee, J., Park, J., Sim, J.: Gscore: Efficient radiance field rendering via architectural support for 3d gaussian splatting. In: ACM International Conference on Architectural Support for Programming Languages and Operating Systems. vol. 3, pp. 497–511 (2024)

35. Li, J., Chen, Z., Wu, X., Wang, L., Wang, B., Zhang, L.: Neural super-resolution for real-time rendering with radiance demodulation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4357–4367 (2024)
36. Li, S., Li, C., Zhu, W., Yu, B., Zhao, Y., Wan, C., You, H., Shi, H., Lin, Y.: Instant-3d: Instant neural radiance field training towards on-device ar/vr 3d reconstruction. In: Annual International Symposium on Computer Architecture. pp. 1–13 (2023)
37. Li, Z., Marshall, C.S., Vembar, D.S., Liu, F.: Future frame synthesis for fast monte carlo rendering. In: Graphics Interface. pp. 74–83 (2022)
38. Lin, L., Liu, Y., Hu, Y., Yan, X., Xie, K., Huang, H.: Capturing, reconstructing, and simulating: The urbanscene3d dataset. In: European Conference on Computer Vision. pp. 93–109 (2022)
39. Liu, E.: Dlss 2.0: Image reconstruction for real-time rendering with deep learning. In: GPU Technology Conference (2020)
40. Liu, Z., Yeh, R.A., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: IEEE/CVF International Conference on Computer Vision. pp. 4463–4471 (2017)
41. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20654–20664 (2024)
42. Mark, W.R., McMillan, L., Bishop, G.: Post-rendering 3d warping. In: Symposium on Interactive 3D Graphics. pp. 7–ff (1997)
43. Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1410–1418 (2015)
44. Niedermayr, S., Neuhauser, C., Westermann, R.: Lightweight gradient-aware upscaling of 3d gaussian splatting images. In: IEEE/CVF International Conference on Computer Vision. pp. 25862–25871 (2025)
45. Niemeyer, M., Manhardt, F., Rakotosaona, M.J., Oechsle, M., Duckworth, D., Gosula, R., Tateno, K., Bates, J., Kaeser, D., Tombari, F.: Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. In: International Conference on 3D Vision. pp. 134–144 (2025)
46. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: IEEE/CVF International Conference on Computer Vision. pp. 261–270 (2017)
47. Qu, H., Li, Z., Rahmani, H., Cai, Y., Liu, J.: Disc-gs: Discontinuity-aware gaussian splatting. Advances in Neural Information Processing Systems **37**, 112284–112309 (2024)
48. Reshetov, A.: Morphological antialiasing. In: Conference on High Performance Graphics. pp. 109–116 (2009)
49. Scherzer, D., Yang, L., Mattausch, O., Nehab, D., Sander, P.V., Wimmer, M., Eisemann, E.: Temporal coherence methods in real-time rendering. In: Computer Graphics Forum. pp. 2378–2408 (2012)
50. Shin, S., Park, J., Cho, S.: Locality-aware gaussian compression for fast and high-quality rendering. arXiv preprint arXiv:2501.05757 (2025)
51. Turki, H., Ramanan, D., Satyanarayanan, M.: Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12922–12931 (2022)
52. Wang, X., Yi, R., Ma, L.: Adr-gaussian: Accelerating gaussian splatting with adaptive radius. In: SIGGRAPH Asia Conference Papers. pp. 1–10 (2024)

53. Wu, M., Dai, H., Yao, K., Tuytelaars, T., Yu, J.: Bg-triangle: Bézier gaussian triangle for 3d vectorization and rendering. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16197–16207 (2025)
54. Wu, R., Liu, Y., Ning, G., Liang, P., Chang, Q.: Ultralight vm-unet: Parallel vision mamba significantly reduces parameters for skin lesion segmentation. Patterns (2024)
55. Wu, S., Kim, S., Zeng, Z., Vembar, D., Jha, S., Kaplanyan, A., Yan, L.Q.: Extrass: A framework for joint spatial super sampling and frame extrapolation. In: SIGGRAPH Asia Conference Papers. pp. 1–11 (2023)
56. Wu, Z., Zuo, C., Huo, Y., Yuan, Y., Peng, Y., Pu, G., Wang, R., Bao, H.: Adaptive recurrent frame prediction with learnable motion vectors. In: SIGGRAPH Asia Conference Papers. pp. 1–11 (2023)
57. Xiao, L., Nouri, S., Chapman, M., Fix, A., Lanman, D., Kaplanyan, A.: Neural supersampling for real-time rendering. ACM Transactions on Graphics **39**(4), 142–1 (2020)
58. Yang, L., Nehab, D., Sander, P.V., Sitthi-Amorn, P., Lawrence, J., Hoppe, H.: Amortized supersampling. ACM Transactions on Graphics **28**(5), 1–12 (2009)
59. Zamfir, E., Conde, M.V., Timofte, R.: Towards real-time 4k image super-resolution. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 1522–1532 (2023)
60. Zhang, B., Fang, C., Shrestha, R., Liang, Y., Long, X., Tan, P.: Rade-gs: Rasterizing depth in gaussian splatting. arXiv preprint arXiv:2406.01467 (2024)
61. Zhang, X., Zeng, H., Zhang, L.: Edge-oriented convolution block for real-time super resolution on mobile devices. In: ACM International Conference on Multimedia. pp. 4034–4043 (2021)
62. Zheng, M., Sun, L., Dong, J., Pan, J.: Efficient video super-resolution for real-time rendering with decoupled g-buffer guidance. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11328–11337 (2025)
63. Zhong, Z., Krishnan, G., Sun, X., Qiao, Y., Ma, S., Wang, J.: Clearer frames, anytime: Resolving velocity ambiguity in video frame interpolation. In: European Conference on Computer Vision. pp. 346–363 (2024)
64. Zhong, Z., Zhu, J., Dai, Y., Zheng, C., Chen, G., Huo, Y., Bao, H., Wang, R.: Fusesr: Super resolution for real-time rendering through efficient multi-resolution fusion. In: SIGGRAPH Asia Conference Papers. pp. 1–10 (2023)
65. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa splatting. IEEE Transactions on Visualization and Computer Graphics **8**(3), 223–238 (2002)