POINTWEAVE: FEED-FORWARD VIEW SYNTHESIS WITH EXPLICIT POINT INTERFACE

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce PointWeave, a feed-forward novel-view renderer that uses explicit 3D points to guide learned appearance reconstruction. Existing 3DGS-based methods render explicit primitives efficiently, but recovering fine detail can require many primitives. Geometric-free methods offer flexible and scalable appearance modeling, but can struggle to maintain geometric consistency under unseen camera transformations. In contrast, PointWeave combines explicit geometry with learned image synthesis. It predicts 3D points with appearance features from input views, and uses learned ray-to-point attention to aggregate their features for each target ray. The resulting feature map is combined with colors retrieved from the input views using the inferred geometry and passed through a decoder to reconstruct the target image. On RealEstate10K, PointWeave achieves 31.38 dB PSNR with 8,192 points, outperforming recent 3DGS-based methods by 1.6 dB while using fewer primitives. It matches the performance of state-of-the-art geometric-free methods with up to 4× faster rendering and substantially better geometric consistency under camera roll, field-of-view, pixel-aspect and world-scale changes. It also outperforms the baselines in zero-shot transfer to unseen datasets, with 0.54–2.02 dB PSNR gains on ACID, DL3DV, ScanNet++, and DTU.

1 INTRODUCTION

Feed-forward novel-view synthesis predicts images from new camera poses given input images, without per-scene optimization (Yu et al., 2021; Wang et al., 2021; Charatan et al., 2024). High-quality synthesis requires both detailed appearance rendering and a reliable relationship between the underlying scene representation and the target camera. Two prominent approaches address these requirements differently. Geometric-based methods predict an explicit 3D scene, commonly represented by 3D Gaussians, and render it with a fixed rasterizer (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025). Geometric-free methods learn the mapping from input images and cameras to target images without an explicit 3D scene representation (Sajjadi et al., 2022; Jin et al., 2025).

3D Gaussian Splatting-based methods couple appearance and geometry in individual primitives. Each Gaussian carries a single view-dependent color, and spatial variation arises through projected footprints and alpha-compositing (Kerbl et al., 2023). Consider a flat wall covered with fine printed text. Its geometry is as simple as that of a uniformly painted wall, but rendering its text generally requires smaller, more numerous Gaussians. The additional primitives serve appearance complexity even though the surface geometry is unchanged. With the fixed rasterizer, 3DGS-based methods can struggle to render fine detail without a large number of primitives.

Geometric-free models predict novel-view images without introducing an explicit 3D scene representation, demonstrating the flexibility of learned appearance reconstruction (Rombach et al., 2021; Sajjadi et al., 2022; Jin et al., 2025; Jiang et al., 2025), and achieve the state-of-the-art rendering quality (Szymanowicz et al., 2026). However, their renderings can lack 3D consistency under camera transformations beyond those encountered during training (Wu et al., 2026b). The relationship between scene features and target cameras depends on the learned renderer and its camera conditioning, motivating efforts to encode relative camera and ray geometry more effectively (Li et al., 2025; Wu et al., 2026a). These observations motivate our central question: *where should geometry enter a feed-forward view-synthesis pipeline, and which operations should it constrain?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our premise is that geometry should guide where a renderer gathers visual evidence and how that evidence is combined. A compact set of 3D points can associate target rays with relevant scene features, while learned appearance features and the input images supply the information needed to reconstruct fine detail. This gives geometry a direct role in rendering while allowing appearance to be modeled beyond the spatial arrangement of primitives.

We introduce **PointWeave**, a feed-forward renderer that implements this idea through geometry-guided ray-to-point attention (Figure 1). From the input images, PointWeave predicts 3D points carrying learned appearance features. For each target ray, we gather K points near the ray and use their relative geometry to weight and combine their view-conditioned features. This produces a feature map aligned with the target view. The inferred geometry also identifies corresponding appearance in the context images. An image decoder combines this observed appearance with the aggregated features to reconstruct the novel view.

This design has two advantages. First, it promotes 3D consistency by grounding target-view synthesis in a shared explicit 3D scene. As the camera changes, each target ray gathers evidence according to its geometric relationship with the same scene points. Reusing the predicted points and their appearance attributes across views also supports efficient rendering. Second, learned point features carry detail beyond the spatial layout of the points, allowing detailed image reconstruction from a compact point representation. Together, these advantages allow PointWeave to achieve better rendering quality than recent 3DGS-based methods with fewer primitives, while matching the performance of state-of-the-art geometric-free methods with up to 4× faster rendering and substantially better geometric consistency.

Our contributions are as follows:

- We introduce PointWeave, a feed-forward novel-view renderer that uses predicted 3D points and geometry-guided ray-to-point attention to select and aggregate visual evidence for learned appearance reconstruction.
- Our method outperforms recent 3DGS-based methods with fewer primitives and matches the performance of state-of-the-art geometric-free methods on RealEstate10K, with stronger geometry consistency and faster rendering speed than the geometric-free methods.
- We demonstrate strong zero-shot generalization from RealEstate10K to ACID, DL3DV, ScanNet++, and DTU, achieving 0.54–2.02 dB PSNR gains over the baselines.

2 Related Work

From scene reconstruction to feed-forward rendering. Neural rendering initially focused on fitting a representation to each scene (Mildenhall et al., 2020; Kerbl et al., 2023). Generalizable methods transfer this inference into networks trained across scenes, combining source-image features with geometric correspondence or ray aggregation (Yu et al., 2021; Wang et al., 2021; Chen et al., 2021; Johari et al., 2022; Suhail et al., 2022; T. et al., 2023). Feed-forward Gaussian methods make the resulting scene explicit and reusable across target cameras (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025; 2026). Within this family, reconstruction accuracy and primitive allocation are complementary concerns: better geometry improves correspondence, while adaptive allocation or compact scene tokens control the cost of representing appearance (Wan et al., 2026; Ren et al., 2026). PointWeave shares the reusable geometric representation, but uses its points to organize learned features for image reconstruction.

Learned rendering and geometric conditioning. Geometric-free synthesis removes the explicit scene representation and learns the image-formation mapping from source images and target-camera inputs (Rombach et al., 2021; Sajjadi et al., 2022; Jin et al., 2025; Jiang et al., 2025). Work on scaling and attention connectivity explores the capacity of this approach (Kim et al., 2026; Jia et al., 2026; Sun et al., 2026). Geometric priors can still enter through pretraining: LagerNVS transfers a reconstruction encoder into a fully neural renderer (Szymanowicz et al., 2026; Wang et al., 2024b; Leroy et al., 2024; Wang et al., 2025a). Relative camera and ray encodings provide another way to structure the mapping (Safin et al., 2023; Miyato et al., 2024; Kong et al., 2024; Li et al., 2025; Wu et al., 2026a). These directions show that geometric knowledge and an explicit rendering representation are separate choices. PVSM makes this distinction concrete by conditioning image synthesis on projections of externally estimated geometry (Wu et al., 2026b). PointWeave

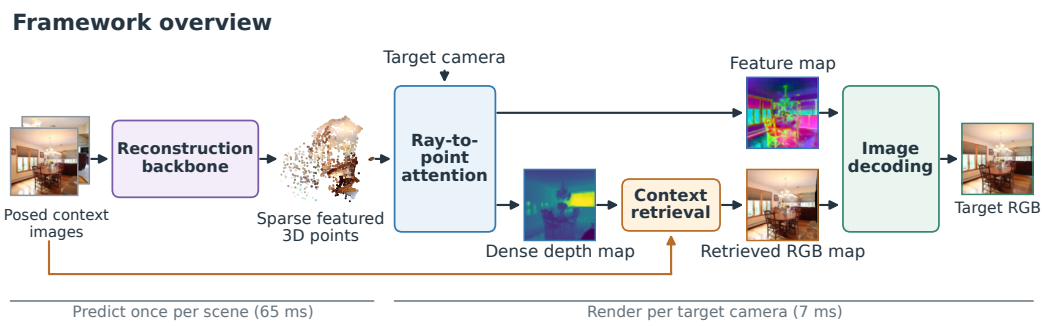

Figure 1: **PointWeave overview.** We use a reconstruction backbone to predict featured 3D points once per scene. Given a target camera, the rays gather features from these points via learned ray-to-point attention. The aggregated features are combined with context appearance retrieved through the predicted geometry to decode the target image. The times are evaluated on an RTX 3090.

also grounds learned rendering in geometry, but jointly predicts the point representation and uses point–ray relations to guide feature aggregation before image decoding.

Explicit features with learned appearance reconstruction. A complementary line of work attach learned features to explicit geometry and use a neural network to turn these features into images (Aliev et al., 2020; Rakhimov et al., 2022; Riegler & Koltun, 2021; Zhu et al., 2024). Point-based methods combine features along viewing rays or learn how points contribute to each ray via attention (Xu et al., 2022; Wang et al., 2024a; Zhang et al., 2023; Chang et al., 2023). Other methods store features in Gaussians and decode the splatted features into images (Martins & Civera, 2024; Chen et al., 2024b; Fu et al., 2026; Lang et al., 2026). PointWeave renders its featured points with ray-to-point attention, which can model complex point–ray relations and thus render fine detail from fewer primitives. Unlike these methods, PointWeave changes both how the representation is obtained and how it is queried. It predicts its points and features in a single feed-forward pass instead of optimizing them per scene or taking them as given, and it makes point attention efficient by factoring each point's appearance into reusable features that are adapted to the viewing direction, rather than evaluating a large MLP or a multi-layer Transformer for every point–ray pair (Zhang et al., 2023; Chang et al., 2023; Ost et al., 2022). Rather than relying on point features alone, its decoder also draws on the context images to recover fine texture beyond what the point attributes store.

3 Method

Given V posed context images $\{(\mathbf{I}_c, \boldsymbol{K}_c, \boldsymbol{E}_c)\}_{c=1}^{V}$ and a target camera $(\boldsymbol{K}_t, \boldsymbol{E}_t)$, PointWeave renders an $H \times W$ image of the target view (Figure 1). Here $\boldsymbol{K}$ and $\boldsymbol{E}$ denote camera intrinsics and extrinsics. PointWeave works in three steps. A reconstruction backbone first predicts a single set of 3D points from all context views, each carrying learned appearance features (Section 3.1). Each target ray then gathers features from the points around it with attention (Section 3.2). Because every ray reads from the same explicit points, renderings stay consistent across viewpoints. Finally, an image decoder turns the gathered features, together with colors looked up in the context images, into the output image (Section 3.3).

3.1 Feed-forward point prediction

We use the ReSplat reconstruction backbone, which builds on DepthSplat (Xu et al., 2026; 2025), to extract multiview features and predict a depth map for each context image. Lifting these depths to 3D on a stride-four pixel grid gives one shared point set $\mathcal{P} = \{(\boldsymbol{p}_j, \boldsymbol{b}_j, \boldsymbol{A}_j, \boldsymbol{n}_j)\}_{j=1}^{M}$ with M points in total across the context views. Each point has a position $\boldsymbol{p}_j$, learned appearance features $(\boldsymbol{b}_j, \boldsymbol{A}_j)$ and the unit direction $\boldsymbol{n}_j$ from its source camera to the point.

A point's appearance can change with the viewing direction, for example on glossy surfaces. Running a deep network for every point–ray pair would capture this, but it repeats much of the same computation across pixels and views. Instead, we split each point's appearance into a part computed once

per point and a small part computed per ray. Two learned linear layers map each point's backbone feature $\boldsymbol{q}_j$ to a base feature $\boldsymbol{b}_j \in \mathbb{R}^C$ and a matrix $\boldsymbol{A}_j \in \mathbb{R}^{C\times R}$, with $C = 32$ and $R = 8$. The base feature holds the appearance that looks the same from every direction. The eight columns of $\boldsymbol{A}_j$ are point-specific feature vectors, and for each ray a small code of eight weights, computed from the viewing geometry (Section 3.2), mixes them to adjust the base feature to the current viewing direction.

3.2 Ray-to-point attention

Each target pixel is rendered from the points around its ray. For ray $r_i = (\boldsymbol{o}_i, \boldsymbol{d}_i)$, with camera center $\boldsymbol{o}_i$ and unit direction $\boldsymbol{d}_i$, we select the K points whose projections lie closest to the pixel in the target image, forming the candidate set $\mathcal{N}_i$ ($K = 20$ by default). Attention then weighs these candidates, so each pixel considers only a small, geometrically relevant set of points.

Geometry plays two roles here: it decides how relevant a candidate is to the ray, and it adjusts the candidate's appearance to the viewing direction. We describe the two with separate inputs. To locate a point relative to a ray, we set up a coordinate frame for each ray, with its origin at $\boldsymbol{o}_i$, its depth axis along $\boldsymbol{d}_i$, and two perpendicular axes $\boldsymbol{e}_i^x, \boldsymbol{e}_i^y$, where $\boldsymbol{e}_i^x$ is orthogonal to the camera's up direction. We transform each point $\boldsymbol{p}_j$ into this local ray coordinate frame and get the point's coordinates $[x_{ij}, y_{ij}, t_{ij}]$, where $x_{ij} = (\boldsymbol{p}_j - \boldsymbol{o}_i)^\top \boldsymbol{e}_i^x$, $y_{ij} = (\boldsymbol{p}_j - \boldsymbol{o}_i)^\top \boldsymbol{e}_i^y$ and depth $t_{ij} = \boldsymbol{d}_i^\top(\boldsymbol{p}_j - \boldsymbol{o}_i)$. With t_i^0 the depth of the nearest candidate in front of the camera, the two inputs are

$$\begin{aligned} \boldsymbol{g}_{ij} &= \big[x_{ij},\ y_{ij},\ t_{ij} - t_i^0\big], \\ \boldsymbol{h}_{ij} &= \big[\boldsymbol{d}_i,\ x_{ij}\boldsymbol{e}_i^x + y_{ij}\boldsymbol{e}_i^y,\ \boldsymbol{n}_j^\top \boldsymbol{d}_i\big]. \end{aligned} \tag{1}$$

The relevance input $\boldsymbol{g}_{ij} \in \mathbb{R}^3$ says where the point sits relative to the ray: how far it lies to the side, and how much deeper it is than the nearest candidate. The appearance input $\boldsymbol{h}_{ij} \in \mathbb{R}^7$ describes how the point is seen: the target viewing direction, the point's sideways offset from the ray, and the cosine between the source and target viewing directions. We apply a sinusoidal positional encoding γ to both. Two separate learned linear layers $f_{\boldsymbol{\theta}}$ and $u_{\boldsymbol{\theta}}$ turn $\gamma(\boldsymbol{g}_{ij})$ into an attention score, and $\gamma(\boldsymbol{h}_{ij})$ into the eight appearance weights. Because the scores depend only on positions measured relative to the ray rather than in world coordinates, they do not tie the model to a particular world frame, which helps it generalize across scenes and viewpoints.

Some regions, such as the sky, may not have reliable predicted points. We therefore add a learned background slot $\varnothing$, which lets attention put weight on the background instead of the selected points. With score $s_{i\varnothing}$ and value $\boldsymbol{v}_\varnothing$ for the background slot, the feature of each ray is

$$\begin{aligned} s_{ij} &= f_{\boldsymbol{\theta}}\big(\gamma(\boldsymbol{g}_{ij})\big), & \boldsymbol{v}_{ij} &= \boldsymbol{b}_j + \boldsymbol{A}_j\, u_{\boldsymbol{\theta}}\big(\gamma(\boldsymbol{h}_{ij})\big), \\ \alpha_{ij} &= \frac{\exp s_{ij}}{\exp s_{i\varnothing} + \sum_{k\in\mathcal{N}_i} \exp s_{ik}}, & \boldsymbol{z}_i &= \sum_{j\in\mathcal{N}_i} \alpha_{ij}\boldsymbol{v}_{ij} + \alpha_{i\varnothing}\boldsymbol{v}_\varnothing. \end{aligned} \tag{2}$$

Here $\alpha_{i\varnothing}$ is the background weight. Collecting $\boldsymbol{z}_i \in \mathbb{R}^C$ over all target pixels gives a feature map $\mathbf{Z} \in \mathbb{R}^{H\times W\times C}$ that is aligned pixel by pixel with the target image.

3.3 Context appearance and image decoding

Summarizing the context images as point features can lose fine texture that is still visible in the images themselves. We therefore let the decoder look up colors directly in the context images, using the predicted geometry to find where to look (Wang et al., 2021; Riegler & Koltun, 2021). For each ray, we reuse the attention weights to estimate where the ray hits the surface:

$$\widehat{d}_i = \frac{\sum_{j\in\mathcal{N}_i} \alpha_{ij} t_{ij}}{\sum_{j\in\mathcal{N}_i} \alpha_{ij} + \epsilon}, \qquad \boldsymbol{x}_i = \boldsymbol{o}_i + \widehat{d}_i \boldsymbol{d}_i.$$

We project $\boldsymbol{x}_i$ into each context camera c and bilinearly sample its color at $\pi_c(\boldsymbol{x}_i)$. This color can be wrong when the point falls outside a context image or is hidden behind another surface, so each sample comes with two visibility signals. A validity mask records whether the projection lies in front of the camera and inside the image. An occlusion signal compares the point's depth with the predicted depth at that context pixel, and a positive log-depth difference means that $\boldsymbol{x}_i$ lies behind the visible

surface. Stacking the colors and these two signals from V_c context views gives $\mathbf{W} \in \mathbb{R}^{H\times W\times 5V_c}$. We use the first and last context views ($V_c = 2$) for more than 2 input views, so $\mathbf{W}$ stays at ten channels.

The decoder must combine the learned point features with context colors that may be missing or misaligned, so we also give it cues about how reliable each ray's attention is. We append five per-pixel values to $\mathbf{Z}$: the estimated depth, the total attention weight on the selected points rather than the background, the entropy of the attention weights, the gap between the two largest point weights, and the perpendicular 3D distance from the highest-weighted point to the ray. This gives $\widetilde{\mathbf{Z}} \in \mathbb{R}^{H\times W\times (C+5)}$. Two separate learned convolution layers map the point features and the context colors into a common space, and we add the results:

$$\mathbf{F}_0 = \phi_Z(\widetilde{\mathbf{Z}}) + \phi_W(\mathbf{W}), \qquad \mathbf{F}_0 \in \mathbb{R}^{H\times W\times 96}. \tag{3}$$

A NAFNet-style image decoder (Chen et al., 2022) then uses the surrounding pixels to produce the final RGB image. We train with an L_1 pixel loss and a perceptual loss:

$$\mathcal{L} = \|\hat{\mathbf{I}} - \mathbf{I}\|_1 + 0.05\,\mathcal{L}_{\text{perc}}(\hat{\mathbf{I}}, \mathbf{I}), \tag{4}$$

where $\mathcal{L}_{\text{perc}}$ is computed on VGG image features (Zhang et al., 2018).

4 Experiments

We compare PointWeave with the baselines on three tasks: 1. **In-domain novel-view synthesis** on RealEstate10K (RE10K) (Zhou et al., 2018) and DL3DV (Ling et al., 2024) (Section 4.1), 2. **Zero-shot cross-dataset transfer** to ACID (Liu et al., 2021), DL3DV, ScanNet++ (Yeshwanth et al., 2023) and DTU (Jensen et al., 2014) (Section 4.2), and 3. **Geometric consistency under camera or world transformations** on the benchmark proposed by PVSM (Wu et al., 2026b) (Section 4.3). We also present ablation studies in Section 4.4 and a geometry-editing application in Section C. We use PSNR, SSIM and LPIPS (Zhang et al., 2018) to evaluate rendering quality quantitatively.

4.1 In-domain novel-view synthesis

We evaluate in-domain novel-view synthesis on RE10K with two input context views and on DL3DV with two, four and six input views. We compare with the 3DGS-based methods pixelSplat, MVSplat, DepthSplat, ReSplat, SplatWeaver and TokenGS (Charatan et al., 2024; Chen et al., 2024a; Xu et al., 2025; 2026; Wan et al., 2026; Ren et al., 2026), the geometric-free methods LVSM, CLiFT, and LagerNVS (Jin et al., 2025; Wang et al., 2025b; Szymanowicz et al., 2026), and the point projection-conditioned method PVSM (Wu et al., 2026b). We include their paper-reported results when available; otherwise, we evaluate their released checkpoints.

Implementation details. We initialize the image encoder and depth predictor from released ReSplat checkpoints and optimize the whole pipeline jointly with a randomly initialized renderer, training separately on each dataset. The RE10K model is trained with a batch size of 16 for 600k steps, with two input views. The DL3DV model is trained with a batch size of 4 for 300k steps, with two to six randomly sampled input views. Each model is trained on eight H100 GPUs. We use AdamW with a learning rate of 2×10^{-4} for the renderer (ray-to-point attention + decoder) and 2×10^{-5} for the ReSplat backbone with cosine decay. The scene and frame sampling strategy follows ReSplat for both training and evaluation. Input images are resized to 256×256 for RE10K and 256×448 for DL3DV. Our standard model uses $K = 20$ nearest points for ray-to-point attention during training and inference. Predicting one point per stride-four grid location gives 4,096 points per view for RE10K and 7,168 points per view for DL3DV. The $K = 5$ variant in Table 1 is fine-tuned for 20k iterations from the model trained with $K = 20$. Inference timings are measured on an RTX 3090 GPU. For DL3DV, as LagerNVS only supports square inputs, we compare it with PointWeave separately at 256×256 for both input and output rather than at our 256×448 training resolution. Following the baselines' respective evaluation protocols, the full-frame block of Table 3 scores all frames, including context views, whereas the square-input block scores only non-context target views.

Results. As shown in Table 1, PointWeave outperforms the compared 3DGS-based methods with fewer primitives (8,192 points) and matches the performance of LagerNVS, a state-of-the-art method

Table 1: **RE10K novel view synthesis.** We use two context views as input and evaluate on 256^2 images. Times are encoding / rendering in ms. †: trained with four context views.

Method	Primitives	Encode/render (ms)	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS-based					
pixelSplat (Charatan et al., 2024)	393k	139.5 / 3.2	25.89	.858	.142
MVSplat (Chen et al., 2024a)	131k	50.5 / 2.0	26.39	.869	.128
DepthSplat (Xu et al., 2025)	131k	118.2 / 3.2	27.47	.889	.114
TokenGS (1,024 tokens) (Ren et al., 2026)	66k	116.8 / 0.6	28.02	.896	.147
TokenGS (4,096 tokens) (Ren et al., 2026)	262k	249.6 / 1.0	28.41	.903	.135
SplatWeaver (Wan et al., 2026)	47k	273.4 / 1.3	29.06	.899	.102
ReSplat (no refinement) (Xu et al., 2026)	131k	105.8 / 1.4	29.40	.909	.104
ReSplat (2 refinements) (Xu et al., 2026)	131k	393.1 / 1.4	29.75	.912	.100
Geometric-free					
LVSM (decoder-only) (Jin et al., 2025)	—	— / 41.9	29.68	.906	.098
CLiFT† (Wang et al., 2025b)	—	875.7 / 10.2	27.06	.867	.132
LagerNVS (Szymanowicz et al., 2026)	—	191.5 / 30.9	**31.39**	.926	**.078**
PointWeave, $K = 20$	8,192	65.6 / 10.8	31.38	**.927**	.093
PointWeave, $K = 5$	8,192	67.7 / 7.0	31.32	.927	.093

Table 2: **Training batch and sample budget on RE10K.** Our method matches the performance of the baselines with $5\times$ fewer training samples, and outperforms them with fewer samples.

Method	Scenes/step	Steps	Training samples	PSNR ↑	SSIM ↑	LPIPS ↓
LVSM (decoder-only)	64	100k	6.4M	28.89	.894	.108
LVSM (decoder-only)	512	100k	51.2M	29.68	.906	.098
LagerNVS	64	100k	6.4M	30.48	.918	.086
LagerNVS	512	100k	51.2M	**31.39**	.926	**.078**
Ours, batch 4	4	300k	1.2M	30.82	.921	.098
Ours, batch 16	16	300k	4.8M	31.12	.925	.095
Ours, batch 16, 600k (full)	16	600k	9.6M	31.38	**.927**	.093

Table 3: **DL3DV novel view synthesis.** We compare with LagerNVS separately as it only supports square inputs. †: trained with four views. ‡: trained with four and six context views.

	2 views			4 views			6 views		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Full-frame: 256×448 ***inputs and outputs***									
MVSplat	17.54	.529	.402	21.63	.721	.233	22.93	.775	.193
DepthSplat	19.31	.615	.310	23.12	.780	.178	24.19	.823	.147
TokenGS†	19.35	.609	.440	23.02	.747	.326	23.69	.766	.311
ReSplat (no refinement)	18.82	.586	.393	23.94	.787	.209	25.79	.840	.162
ReSplat (2 refinements)	19.55	.603	.377	25.28	.813	.186	27.30	.867	.139
CLiFT‡	18.93	.573	.368	24.58	.774	.190	26.52	.825	.147
PointWeave (ours)	**23.36**	**.730**	**.263**	**28.81**	**.876**	**.126**	**31.30**	**.920**	**.087**
Square-input: 256^2 ***inputs and outputs***									
LagerNVS	**23.30**	**.739**	**.203**	**27.56**	**.869**	**.095**	29.45	.904	**.068**
PointWeave (ours)	21.95	.696	.285	27.42	.861	.135	**29.96**	**.911**	.090

on RE10K. On an RTX 3090, PointWeave encodes a scene in 65 ms and renders each target in 10.8 ms at $K = 20$ or 7.0 ms at $K = 5$, around 3–4$\times$ faster than LagerNVS. Figure 2 shows qualitative results on RE10K. PointWeave produces more detailed images than the 3DGS-based baselines, particularly in challenging regions containing thin structures and complex geometry, which require densely distributed Gaussians for accurate rendering. Geometry-free baselines may hallucinate structures, as illustrated by the extra window frame in the bottom example of Figure 2.

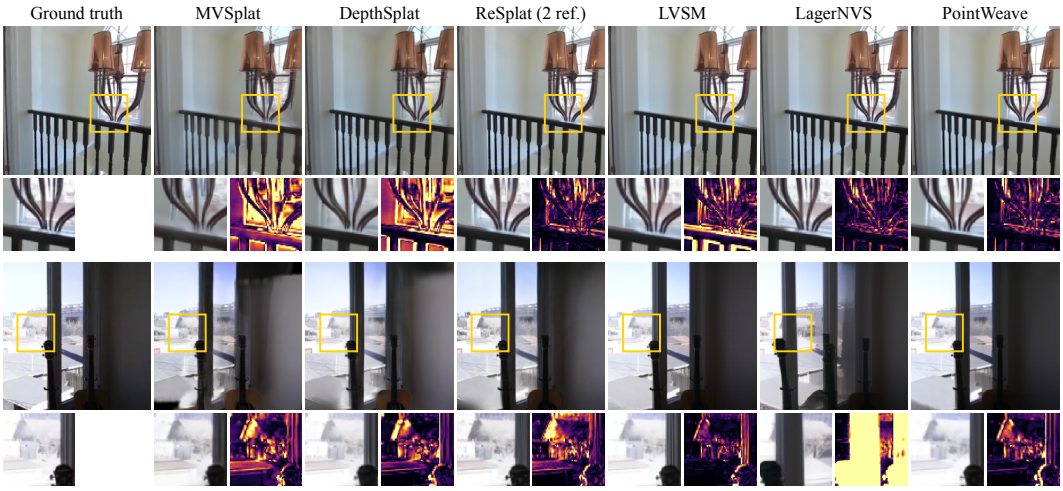

Figure 2: **RE10K, two context views.** We show the target, a $4\times$ crop (yellow box) and the crop's error map, where brighter means larger error. 3DGS-based methods blur when the point density is low. Geometric-free methods may hallucinate geometry, such as the extra window frame (bottom).

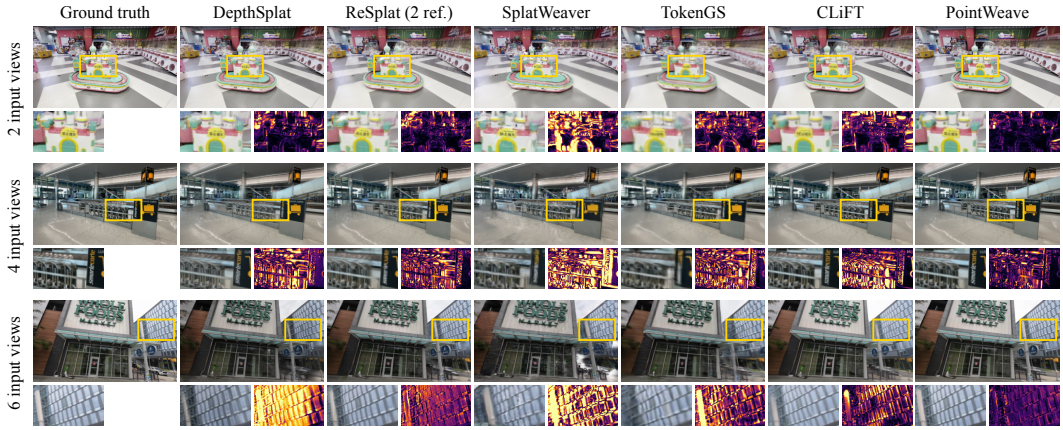

Figure 3: **DL3DV, full-frame protocol, two to six context views**, in the layout of Figure 2. Our method retains more detail than the baselines with fewer primitives and better geometric consistency.

Table 2 shows that PointWeave outperforms LVSM and matches LagerNVS with $5\times$ fewer training samples: 9.6M versus 51.2M. With 4.8M samples, PointWeave also achieves higher PSNR than both baselines trained with 6.4M samples.

Table 3 compares novel-view synthesis on DL3DV. We compare with the baselines that have released checkpoints trained on DL3DV. PointWeave outperforms the compared 3DGS-based methods under the full-frame protocol by a large margin. Under the square-input protocol, LagerNVS leads with two and four context views, while PointWeave achieves higher PSNR with six views: 29.96 versus 29.45 dB. Figure 3 shows full-frame comparisons, and Figure 4 compares PointWeave with LagerNVS under the square-input protocol. As shown, our method preserves more detail than the 3DGS-based baselines and matches the visual quality of LagerNVS with four and six context views. With two context views, however, PointWeave still falls behind LagerNVS, likely for two reasons. LagerNVS builds on VGGT (Wang et al., 2025a), a large reconstruction model pretrained on broad 3D data, which may provide strong prior knowledge of scene structure. In addition, every PointWeave point is lifted from the predicted depth of a context view, so the points only cover surfaces seen in the context views. Our model can therefore reproduce observed regions faithfully but may struggle to fill in regions that no context view sees. These unobserved regions are largest with two views, which is consistent with the gap narrowing as more views are added.

4.2 Zero-shot cross-dataset novel-view synthesis

We test our RE10K-trained checkpoints on ACID (Liu et al., 2021), DL3DV, ScanNet++ (Yeshwanth et al., 2023) and DTU (Jensen et al., 2014), without target-dataset adaptation. For zero-shot transfer to DL3DV, we use a shorter test-frame gap than in the in-domain evaluation for all methods, following DepthSplat (Xu et al., 2025). For all datasets, we follow the camera pose normalization used in LagerNVS (Szymanowicz et al., 2026). As shown in Table 4, PointWeave exceeds the strongest compared baseline in PSNR by 0.54 dB on ACID, 1.75 dB on DL3DV, 2.02 dB on ScanNet++ and 1.78 dB on DTU. These results demonstrate strong zero-shot generalization across the evaluated datasets. Figure 5 shows examples on ACID, DL3DV, ScanNet++ and DTU.

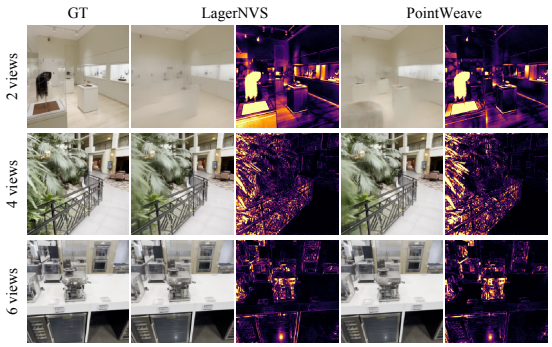

Figure 4: **PointWeave and LagerNVS, square-input DL3DV**, with 2, 4, 6 input views.

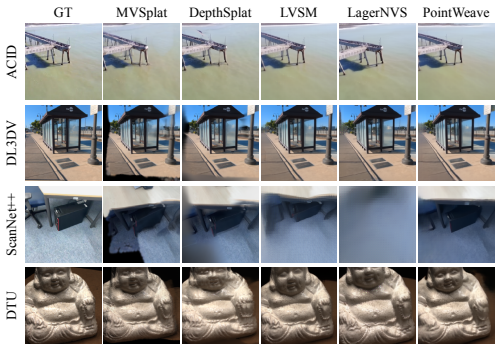

Figure 5: **Zero-shot transfer**. Our method shows better geometric consistency.

Table 4: **Zero-shot transfer from RE10K,** 256^2, two context views. All the baselines are trained on RE10K only. †: trained with four context views.

	ACID (1,595)			DL3DV short (139)			ScanNet++ (50)			DTU (64)		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS-based												
pixelSplat	27.82	.835	.155	27.20	.882	.103	18.42	.720	.278	11.53	.327	.633
MVSplat	28.16	.841	.147	26.95	.873	.101	17.14	.687	.297	13.94	.474	.386
DepthSplat	28.38	.848	.142	28.14	.905	.083	20.79	.761	.254	14.59	.425	.437
ReSplat (no refinement)	29.69	.865	.137	30.53	.927	.075	22.82	.787	.237	15.33	.620	.389
Geometric-free												
LVSM (decoder-only)	30.44	.869	.127	30.03	.914	.080	25.49	.785	.212	16.01	.544	.343
CLiFT†	29.00	.840	.164	26.87	.850	.134	23.09	.760	.269	14.25	.492	.533
LagerNVS	31.70	.886	**.110**	29.72	.907	**.066**	25.30	.785	**.194**	14.75	.498	.375
PointWeave (ours)	**32.25**	**.896**	.124	**32.28**	**.939**	.069	**27.51**	**.829**	.199	**17.79**	**.684**	**.315**

4.3 Geometric consistency under camera transformations

Following PVSM (Wu et al., 2026b), we evaluate the 3D consistency of our method and the baselines under camera or world transformations, including camera roll, field of view, pixel anisotropy and world scale. The test set contains 100 RE10K scenes with three targets each. For each transformation type, we evaluate multiple settings, such as different rotation angles or scale factors, and average the scores across those settings. Following PVSM's metric code, both PSNR and SSIM are computed over valid pixels.

As shown in Table 5 and Figure 6, without any augmentation, PointWeave retains better geometric consistency than the baselines under all four transformation sweeps, including the released 24-layer PVSM checkpoint fine-tuned with additional augmentations.

4.4 Ablation studies

We examine the effects of decoder architecture and the proposed context retrieval on both novel-view synthesis quality and geometric consistency on RE10K, using the protocols in Sections 4.1 and 4.3.

Table 5: **Consistency benchmark.** Following PVSM, we evaluate on RE10K under camera roll, field of view, pixel anisotropy and world scale, given two context views. †: released model fine-tuned with additional camera augmentations. ‡: trained with four context views.

	Roll		FOV		Anisotropy		Scale	
Method	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
LagerNVS	17.72	.480	14.05	.801	16.02	.758	15.22	.512
LVSM (decoder-only)	18.03	.488	16.44	.847	18.93	.812	20.63	.645
CLiFT‡	17.95	.494	17.07	.872	17.82	.784	18.34	.548
PVSM†	19.96	.709	20.17	.928	19.23	.842	21.70	.729
PointWeave (ours)	**21.85**	**.790**	**21.67**	**.953**	**24.41**	**.932**	**25.20**	**.805**

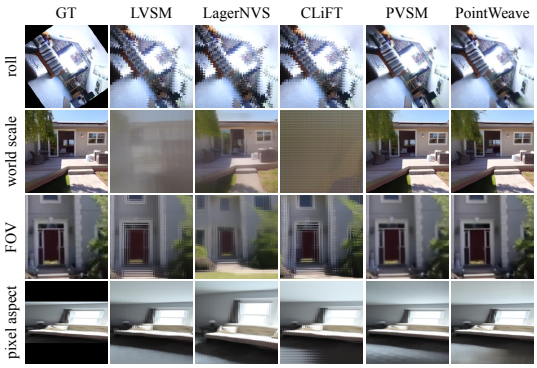

Figure 6: **Renders under camera transformations**. Geometric-free methods fail to render geometry consistent images under transformations.

Table 6: **Renderer ablations** on RE10K. Params.: number of decoder-head parameters. †: decoder head only. Cons.: mean PSNR over the four axes in the consistency benchmark.

Variant	Params.	PSNR ↑	SSIM ↑	LPIPS ↓	ms ↓	Cons. ↑
(a) Decoder						
MLP small	7.4k	28.28	.891	.158	0.2†	26.36
MLP large	49.9M	28.69	.901	.130	86.5†	26.77
NAFnet (ours)	15.2M	30.76	.920	.099	3.2†	25.12
NAFnet large	49.9M	30.02	.914	.111	8.3†	24.87
DPT	54.1M	29.21	.899	.149	5.1†	24.80
Transformer	44.9M	28.87	.896	.148	2.3†	25.05
(b) Context						
Off		30.52	.921	.099	—	23.58
On		30.82	.921	.098	—	24.48

Decoder type. We replace the image decoder of a trained model with different randomly initialized decoders and fine-tune the entire model for 20k steps. For the DPT and Transformer heads, we adopt standard architectures from the literature (Jin et al., 2025; Wu et al., 2026b; Ranftl et al., 2021; Vaswani et al., 2017; Dosovitskiy et al., 2021; Su et al., 2024). For the MLP and NAFNet heads, we evaluate two variants each: one with a parameter count comparable to those of the DPT and Transformer heads, and one using a standard configuration. As shown in Table 6a, the MLP head performs best on the consistency benchmark, as it avoids dependencies on neighboring pixels. NAFNet achieves the best rendering quality with a modest reduction in geometric consistency.

Context retrieval. We evaluate the effect of the context retrieval using variants of our model trained from scratch for 300k steps, with and without the retrieval. As shown in Table 6b, the context retrieval improves PSNR by 0.30 dB, while also improving geometric consistency by 0.90 dB on average across the four transformations in the consistency benchmark.

5 Conclusion and Limitations

PointWeave uses explicit points to organize rendering while learning appearance aggregation and image decoding. It achieves better rendering quality than 3DGS-based methods with fewer primitives, and outperforms or matches the performance of the state-of-the-art geometric-free methods with faster rendering and better geometric consistency. It also shows better zero-shot generalization across datasets than the baselines. There is still a gap between PointWeave and the state-of-the-art methods under sparse context views on DL3DV, which could potentially be reduced by switching the backbone to more powerful 3D foundation models such as VGGT (Wang et al., 2025a). Currently, PointWeave is limited to static scenes, and future work could extend it to dynamic scenes by modeling how the points move over time. Beyond novel view synthesis, our method could also serve as a 3D memory for world models or camera-controlled video generation, so that regions revisited by the camera are rendered from the same points and stay consistent.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, design and provide feedback on experiments, clean and reformat datasets, and interpret results. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims or assist with their proofs, or propose or refine hypotheses, and translation and qualitative or thematic data analysis are not applicable to this work. We have reviewed all AI-assisted work including the training and evaluation scripts. All results for our method and for the baselines are either taken from the original papers or re-evaluated ourselves using their released checkpoints, as listed in Section B. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor S. Lempitsky. Neural Point-Based Graphics. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, pp. 696–712, 2020. URL https://doi.org/10.1007/978-3-030-58542-6_42.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment Anything with Concepts. *CoRR*, abs/2511.16719, 2025. URL https://doi.org/10.48550/arXiv.2511.16719.

Jen-Hao Rick Chang, Wei-Yu Chen, Anurag Ranjan, Kwang Moo Yi, and Oncel Tuzel. Pointersect: Neural Rendering with Cloud-Ray Intersection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 8359–8369, 2023. URL https://doi.org/10.1109/CVPR52729.2023.00808.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 19457–19467, 2024. URL https://doi.org/10.1109/CVPR52733.2024.01840.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 14104–14113, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01386.

Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple Baselines for Image Restoration. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part VII*, pp. 17–33, 2022. URL https://doi.org/10.1007/978-3-031-20071-7_2.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-view Images. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXI*, pp. 370–386, 2024a. URL https://doi.org/10.1007/978-3-031-72664-4_21.

Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. MVSplat360: Feed-Forward 360 Scene Synthesis from Sparse Views. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024b. URL http://papers.nips.cc/paper_files/paper/2024/hash/c196239c5f9481e0db2755f31fe4585f-Abstract-Conference.html.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Guangming Fu, Jiahui Fan, Jian Yang, Milos Hasan, and Beibei Wang. F-RNG: Feed-Forward Relightable Neural Gaussians. *CoRR*, abs/2605.25975, 2026. URL https://doi.org/10.48550/arXiv.2605.25975.

Rasmus Ramsbøl Jensen, Anders Lindbjerg Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large Scale Multi-view Stereopsis Evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 406–413, 2014. URL https://doi.org/10.1109/CVPR.2014.59.

Xiaosong Jia, Yihang Sun, Junqi You, Songbur Wong, Zichen Zou, Junchi Yan, Zuxuan Wu, and Yu-Gang Jiang. Efficient-LVSM: Faster, Cheaper, and Better Large View Synthesis Model via Decoupled Co-Refinement Attention. *CoRR*, abs/2602.06478, 2026. URL https://doi.org/10.48550/arXiv.2602.06478.

Hanwen Jiang, Hao Tan, Peng Wang, Hai Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, and Georgios Pavlakos. RayZer: A Self-supervised Large View Synthesis Model. In *IEEE/CVF International Conference on Computer Vision, ICCV 2025, Honolulu, HI, USA, October 19-25, 2025*, pp. 4918–4929, 2025. URL https://doi.org/10.1109/ICCV51701.2025.00468.

Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025. URL https://openreview.net/forum?id=QQBPWtvtcn.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing NeRF with Geometry Priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 18344–18347, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01782.

Nikhil Varma Keetha, Norman Müller, Johannes L. Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian A. Scherer, and Peter Kontschieder. MapAnything: Universal Feed-Forward Metric 3D Reconstruction. In *International Conference on 3D Vision, 3DV 2026, Vancouver, BC, Canada, March 20-23, 2026*, pp. 499–509, 2026. URL https://doi.org/10.1109/3DV69130.2026.00054.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023. URL https://doi.org/10.1145/3592433.

Evan Kim, Hyunwoo Ryu, Thomas W. Mitchel, and Vincent Sitzmann. Scaling View Synthesis Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pp. 28893–28902, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Kim_Scaling_View_Synthesis_Transformers_CVPR_2026_paper.html.

Xin Kong, Shikun Liu, Xiaoyang Lyu, Marwan Taher, Xiaojuan Qi, and Andrew J. Davison. Eschernet: A Generative Model for Scalable View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 9503–9513, 2024. URL https://doi.org/10.1109/CVPR52733.2024.00908.

Xiaolei Lang, Ze Kang, Zehao Huang, and Naiyan Wang. RoGe: Novel View Synthesis via End-to-End Implicit Reconstruction and Generation, 2026. URL https://arxiv.org/abs/2609.02847.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding Image Matching in 3D with MASt3R. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXII*, pp. 71–91, 2024. URL https://doi.org/10.1007/978-3-031-73220-1_5.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as Relative Positional Encoding. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/17a7075094632c88cccdd86270ad715b-Abstract-Conference.html.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A Large-Scale Scene Dataset for Deep Learning-based 3D Vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 22160–22169, 2024. URL https://doi.org/10.1109/CVPR52733.2024.02092.

Andrew Liu, Ameesh Makadia, Richard Tucker, Noah Snavely, Varun Jampani, and Angjoo Kanazawa. Infinite Nature: Perpetual View Generation of Natural Scenes from a Single Image. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 14438–14447, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01419.

Tomás Berriel Martins and Javier Civera. Feature Splatting for Better Novel View Synthesis with Low Overlap. In *35th British Machine Vision Conference, BMVC 2024, Glasgow, UK, November 25-28, 2024*, 2024. URL https://bmvc2024.org/proceedings/207/.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, pp. 405–421, 2020. URL https://doi.org/10.1007/978-3-030-58452-8_24.

Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. GTA: A Geometry-Aware Attention Mechanism for Multi-View Transformers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024. URL https://openreview.net/forum?id=uJVHygNeSZ.

Julian Ost, Issam H. Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural Point Light Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 18398–18408, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01787.

Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. NPBG++: Accelerating Neural Point-Based Graphics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 15948–15958, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01550.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 12159–12168, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01196.

Jiawei Ren, Michal J. Tyszkiewicz, Jiahui Huang, and Zan Gojcic. TokenGS: Decoupling 3D Gaussian Prediction from Pixels with Learnable Tokens. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026*, 2026.

Gernot Riegler and Vladlen Koltun. Stable View Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 12216–12225, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Riegler_Stable_View_Synthesis_CVPR_2021_paper.html.

Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-Free View Synthesis: Transformers and no 3D Priors. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 14336–14346, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01409.

Aleksandr Safin, Daniel Duckworth, and Mehdi S. M. Sajjadi. RePAST: Relative Pose Attention Scene Representation Transformer. *CoRR*, abs/2304.00947, 2023. URL https://doi.org/10.48550/arXiv.2304.00947.

Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas A. Funkhouser, and Andrea Tagliasacchi. Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 6219–6228, 2022. URL https://doi.org/10.1109/CVPR52688.2022.00613.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. URL https://doi.org/10.1016/j.neucom.2023.127063.

Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable Patch-Based Neural Rendering. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII*, pp. 156–174, 2022. URL https://doi.org/10.1007/978-3-031-19824-3_10.

Cheng Sun, Jaesung Choe, Min-Hung Chen, Ryo Hachiuma, and Yu-Chiang Frank Wang. DVSM: Decoder-only View Synthesis Model Done Right. *CoRR*, abs/2605.29891, 2026. URL https://doi.org/10.48550/arXiv.2605.29891.

Stanislaw Szymanowicz, Minghao Chen, Jianyuan Wang, Christian Rupprecht, and Andrea Vedaldi. LagerNVS: Latent Geometry for Fully Neural Real-time Novel View Synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2026.

Mukund Varma T., Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is Attention All That NeRF Needs? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. URL https://openreview.net/forum?id=xE-LtsE-xx.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Yecong Wan, Fan Li, Mingwen Shao, and Wangmeng Zuo. SplatWeaver: Learning to Allocate Gaussian Primitives for Generalizable Novel View Synthesis. *CoRR*, abs/2605.07287, 2026. URL https://doi.org/10.48550/arXiv.2605.07287.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotný. VGGT: Visual Geometry Grounded Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 5294–5306, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VGGT_Visual_Geometry_Grounded_Transformer_CVPR_2025_paper.html.

Jiaxu Wang, Ziyi Zhang, and Renjing Xu. Learning Robust Generalizable Radiance Field with Visibility and Feature Augmented Point Representation. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024a. URL https://openreview.net/forum?id=o4CLLlIaaH.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 4690–4699, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_IBRNet_Learning_Multi-View_Image-Based_Rendering_CVPR_2021_paper.html.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 20697–20709, 2024b. URL https://doi.org/10.1109/CVPR52733.2024.01956.

Zhengqing Wang, Yuefan Wu, Jiacheng Chen, Fuyang Zhang, and Yasutaka Furukawa. CLiFT: Compressive Light-Field Tokens for Compute Efficient and Adaptive Neural Rendering. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b. URL http://papers.nips.cc/paper_files/paper/2025/hash/0cdc4f2263989c9591e7017ecbdfc30d-Abstract-Conference.html.

Yu Wu, Minsik Jeon, Jen-Hao Rick Chang, Oncel Tuzel, and Shubham Tulsiani. RayRoPE: Projective Ray Positional Encoding for Multi-view Attention. In *European Conference on Computer Vision (ECCV)*, 2026a.

Zirui Wu, Zeren Jiang, Martin R. Oswald, and Jie Song. From Rays to Projections: Better Inputs for Feed-Forward View Synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pp. 29012–29022, 2026b. URL https://openaccess.thecvf.com/content/CVPR2026/papers/Wu_From_Rays_to_Projections_Better_Inputs_for_Feed-Forward_View_Synthesis_CVPR_2026_paper.pdf.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian Splatting and Depth. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 16453–16463, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xu_DepthSplat_Connecting_Gaussian_Splatting_and_Depth_CVPR_2025_paper.html.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. ReSplat: Learning Recurrent Gaussian Splatting. In *Computer Vision - ECCV 2026 - 19th European Conference, Malmö, Sweden, September 8-12, 2026, Proceedings, Part XXXIX*, pp. 364–382, 2026. URL https://doi.org/10.1007/978-3-032-37435-6_20.

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based Neural Radiance Fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 5428–5438, 2022. URL https://doi.org/10.1109/CVPR52688.2022.00536.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/26cfdcd8fe6fd75cc53e92963a656c58-Abstract-Conference.html.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 12–22, 2023. URL https://doi.org/10.1109/ICCV51070.2023.00008.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields From One or Few Images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 4578–4587, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Yu_pixelNeRF_Neural_Radiance_Fields_From_One_or_Few_Images_CVPR_2021_paper.html.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 586–595, 2018. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Yanshu Zhang, Shichong Peng, Alireza Moazeni, and Ke Li. PAPR: Proximity Attention Point Rendering. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/bda5c35eded86adaf0231748e3ce071c-Abstract-Conference.html.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point Transformer. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 16239–16248, 2021. URL https://doi.org/10.1109/ICCV48922.2021.01595.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis using Multiplane Images. *ACM Trans. Graph.*, 37(4):65, 2018. URL https://doi.org/10.1145/3197517.3201323.

Qingtian Zhu, Zizhuang Wei, Zhongtian Zheng, Yifan Zhan, Zhuyu Yao, Jiawang Zhang, Kejian Wu, and Yinqiang Zheng. RPBG: Towards Robust Neural Point-Based Graphics in the Wild. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XV*, pp. 389–406, 2024. URL https://doi.org/10.1007/978-3-031-72633-0_22.

A Implementation Details

Predicting points. Our encoder builds on ReSplat (Xu et al., 2026), which extends the DepthSplat backbone (Xu et al., 2025). DepthSplat predicts depth for each input view from two branches. A multi-view branch matches features across views in a plane-sweep cost volume (Chen et al., 2024a), and a monocular branch uses features from a pretrained Depth Anything V2 ViT-B encoder (Yang et al., 2024; Dosovitskiy et al., 2021). A U-Net combines the two branches and regresses depth. ReSplat then adds a point transformer that refines the features of the 3D points lifted from these depths. It has 12 attention blocks that alternate between local k-nearest-neighbor attention over 16 neighboring points (Zhao et al., 2021) and global attention over the points of all views (Vaswani et al., 2017). We predict a depth for every fourth pixel in each direction and convert these depths to 3D points. For pixel $\boldsymbol{u}_j$ in input view c, the point is

$$\boldsymbol{p}_j = \boldsymbol{t}_c + \boldsymbol{R}_c\big(\boldsymbol{D}_c(\boldsymbol{u}_j)\boldsymbol{K}_c^{-1}\widetilde{\boldsymbol{u}}_j\big), \tag{5}$$

where $\boldsymbol{E}_c = [\boldsymbol{R}_c \,|\, \boldsymbol{t}_c]$ gives the camera's rotation and position, $\boldsymbol{K}_c$ holds its intrinsic parameters, and $\widetilde{\boldsymbol{u}}_j$ is the pixel coordinate with a 1 appended. The depth $\boldsymbol{D}_c(\boldsymbol{u}_j)$ is measured along the camera's viewing axis. With V input images of size $H \times W$, this gives $M = V(H/4)(W/4)$ points: 4,096 per 256^2 image (8,192 for two images), and 14,336, 28,672 and 43,008 points for two, four and six images at 256×448. Each point stores a 32-value feature $\boldsymbol{b}_j$, a 32×8 matrix $\boldsymbol{A}_j$ for view-dependent appearance, and its source viewing direction $\boldsymbol{n}_j$. The feature and matrix together hold 288 values.

Selecting points. We project all points into the target camera and discard those outside its viewing frustum. For each pixel, a search over an image grid finds the $K = 20$ points that project closest to it. A selected point can still be far from the ray in 3D, for example when it lies at a very different depth. Attention then uses each point's 3D position in the local ray coordiantes to decide how much it contributes.

Computing attention and appearance. For both $\boldsymbol{g}_{ij}$ and $\boldsymbol{h}_{ij}$, we keep the original values and add sines and cosines at six frequencies: $\gamma(\boldsymbol{x}) = [\boldsymbol{x}, \{\sin(2^\ell \boldsymbol{x}), \cos(2^\ell \boldsymbol{x})\}_{\ell=0}^{5}]$. A learned linear layer of width 128, shared by all points, computes the attention score s_{ij}. The vector $\boldsymbol{h}_{ij}$ also records how well the target ray agrees with the point's source direction $\boldsymbol{n}_j$. A learned linear layer with a bias, $u_{\boldsymbol{\theta}}$, maps its 91 encoded values to the eight appearance weights. Each point's base feature and 32×8 matrix are computed once and reused for every target view.

Five extra values per pixel. Besides the aggregated features, the decoder receives five values that describe the depth and how the attention weights are spread. Let d_i be the attention-weighted depth, a_i the total weight given to points rather than the background, and e_i the normalized entropy of the weights, which is high when the weight is spread over many points. Let m_i be the gap between the two largest point weights and q_i the perpendicular 3D distance from the highest-weighted point to the ray. The five values are

$$\left[\log\left(\frac{\max(d_i, \epsilon)}{\mu}\right),\ a_i,\ e_i,\ m_i,\ \log(1 + q_i/\mu)\right], \qquad \mu = \max\big(\text{median}_i \,|\, \text{stopgrad}(d_i)|, \epsilon\big). \tag{6}$$

Here, ϵ is a small positive constant, and μ is the median absolute depth over the image, which rescales depths and distances. Gradients do not flow through μ.

Retrieving colors from input images. The attention-weighted depth $\widehat{d}_i$ gives a 3D location $\boldsymbol{x}_i = \boldsymbol{o}_i + \widehat{d}_i \boldsymbol{d}_i$ on the ray. Projecting this location into each selected input image gives five values:

$$\boldsymbol{w}_{ic} = \left[\mathbf{I}_c(\pi_c(\boldsymbol{x}_i)),\ \nu_{ic},\ \tanh\left(\frac{\log(z_c(\boldsymbol{x}_i)/\widehat{\boldsymbol{D}}_c(\pi_c(\boldsymbol{x}_i)))}{\tau}\right)\right], \tag{7}$$

namely the interpolated RGB color, a flag ν_{ic} that marks whether the projection is valid, and a depth comparison that helps detect occlusion. Here, $z_c(\boldsymbol{x}_i)$ is the depth of $\boldsymbol{x}_i$ along the input camera's viewing axis, $\widehat{\boldsymbol{D}}_c$ is that camera's predicted depth map, and $\tau = 0.08$. The weighted depth is divided by the total point weight, so it becomes unreliable when this weight is close to zero. The validity flag only checks the projection, so the decoder also receives the total weight (a_i above) to recognize such cases. No gradients flow from this step into the sampling depth or the input depth maps.

Producing the image. The decoder is a NAFNet (Chen et al., 2022) of width 96, with $(1, 1, 2)$ blocks at its encoder levels, two middle blocks and $(1, 1, 1)$ blocks at its decoder levels. Its input layer ϕ_Z in Equation (3) is a single 3×3 convolution on the 37 channels of $\widetilde{\mathbf{Z}}$, the 32 feature channels and the five extra values. ϕ_W is a second single 3×3 convolution on the ten retrieved channels of $\mathbf{W}$, five from each of the two input views, and the two outputs are added. The last convolution of the decoder outputs three values per pixel, and a sigmoid function maps them to RGB colors between 0 and 1. Before this sigmoid, we also add the average retrieved color over the views with a valid projection. This color is first converted to the scale before the sigmoid (by the inverse sigmoid) and multiplied by a learned scalar weight, and pixels with no valid projection receive no such term. This passes observed colors directly to the output.

B Evaluation Protocols

Test sets. For in-domain evaluation, the RE10K test set has 6,474 cases from the pixelSplat index, each with two input views and three target views. The DL3DV test set has 139 scenes from the DepthSplat split, with two, four or six input views taken from the first 50 frames of each scene. For zero-shot transfer, we use 1,595 ACID cases, 50 ScanNet++ cases with three targets each, and 64 DTU cases from 16 scans. For DL3DV, we use 139 cases from the DepthSplat index in which the two input frames and four target frames all come from the first ten frames of each scene. The views are therefore much closer together than in the in-domain DL3DV test, where they spread over 50 frames.

Baseline settings. Some baseline scores are taken from the original papers. In Table 1, these are pixelSplat, MVSplat, DepthSplat, TokenGS, SplatWeaver and ReSplat with two refinement steps. In Table 2, these are LVSM and LagerNVS trained with 64 scenes per step. In Table 3, these are the four- and six-view scores of LagerNVS (Szymanowicz et al., 2026). We evaluate all other baseline results ourselves with the released checkpoints, including the two-view result of LagerNVS on DL3DV.

Camera and world transformations. Roll, field of view and pixel aspect ratio follow the settings of PVSM's benchmark generator. The camera is rolled by 160 or 205 degrees. The target's horizontal field of view is set to 98, 126 or 173 degrees on the original 640×360 frame, so every setting zooms out. Pixel aspect ratios f_x/f_y are 0.1 and 0.25, split evenly between the two focal lengths. World-scale factors are $0.01, 0.1, 0.5, 2, 10$ and 100. All methods are tested on the same 100 cases with identical input images.

For PVSM, we use the released 24-layer model with depth from a frozen MapAnything model (Keetha et al., 2026), following the suggestion of the PVSM authors. The evaluated model includes 500 extra training steps with camera augmentation, while PointWeave uses no camera augmentation.

C Zero-shot Scene Editing

PointWeave's explicit points let users edit a scene without any additional training. The interface uses SAM 3 (Carion et al., 2025) to turn user clicks into object masks and then selects the points inside those regions. Figure 7 shows sofa scaling and rotation, vase removal and vase duplication, each shown from two views.

Preserving appearance during editing. For a rigid edit $\boldsymbol{T}$, we apply $\boldsymbol{T}^{-1}$ before retrieving input colors, so moved points still take their colors from where they were originally observed. Visibility is checked against the original geometry, and the points' viewing directions move with them. Content that was never observed and the boundaries of edited regions can still cause errors.

D Additional Experiments and Examples

Number of points. We compare 8,192 points with 131,072 points on RE10K with two input views, obtained by learning to predict one point per pixel rather than per 4×4 grid. Both models start from the same weights and train for 300k steps with four scenes per step, the same loss and color retrieval from the input images. On the full RE10K test set, PSNR changes from 30.771 to 30.732 dB

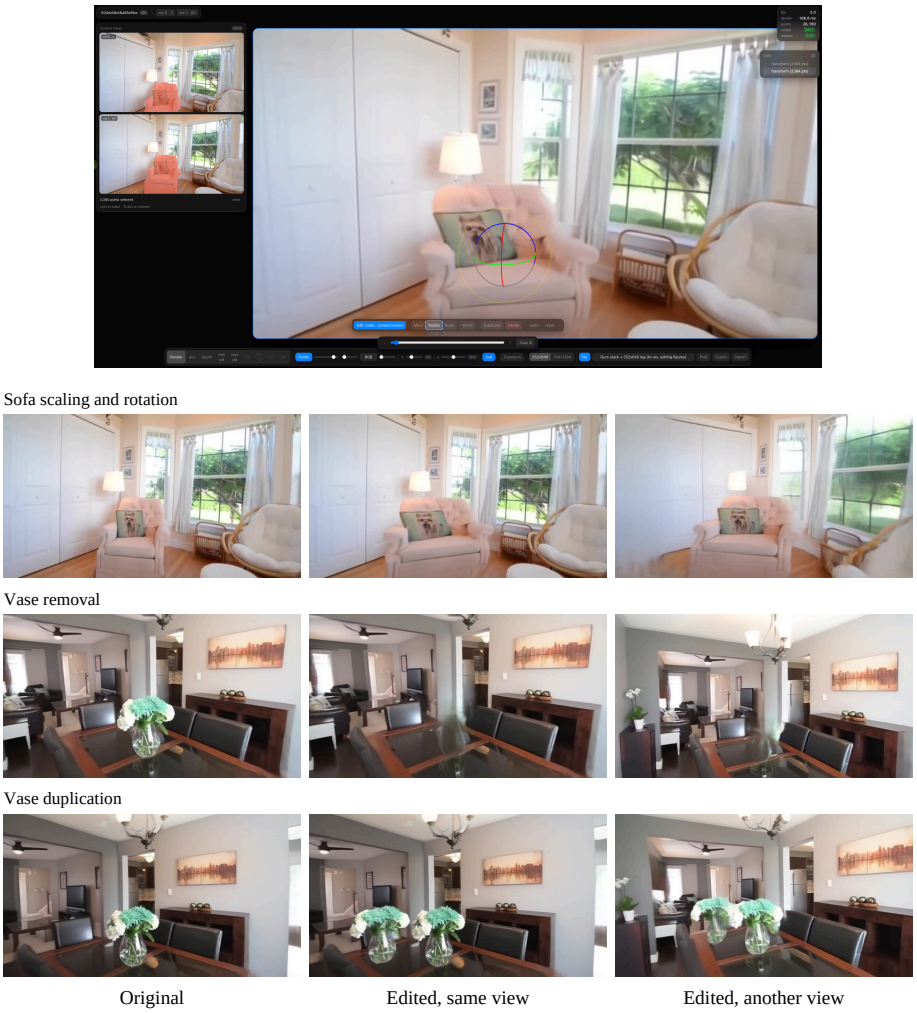

Figure 7: **Scene editing without additional training.** Top: the editing interface, with the selected sofa highlighted. Below: sofa scaling and rotation, vase removal and vase duplication. Each row shows the original scene, the edited scene from the same view and another view.

and LPIPS from 0.0875 to 0.0882, while decoding takes 48% longer. More points therefore do not improve quality here. A likely reason is that the point transformer in the ReSplat backbone was pretrained with the sparser points, one per 4×4 grid, so the denser points change the neighborhoods its attention was trained on.

Additional rendered examples. Figures 8 and 9 show more RE10K and DL3DV scenes.

E Consistency of 3DGS-Based Methods and Camera Augmentation

3DGS-based methods predict an explicit set of Gaussians, so a transformation can be applied to the Gaussians directly instead of through the network. We predict the Gaussians once from the original input views, transform them in the target camera's frame, and render them with the original target camera. We evaluate two variants: transforming only the Gaussian means, and transforming both the means and the covariances. Note that the second variant is harder to use in practice, because it requires the ground-truth transformation field, which is usually not available. Transforming both the means and the covariances (Σ) gives exactly the same image as rendering with the transformed camera. World scaling then leaves the image unchanged, so the scale score equals the untransformed

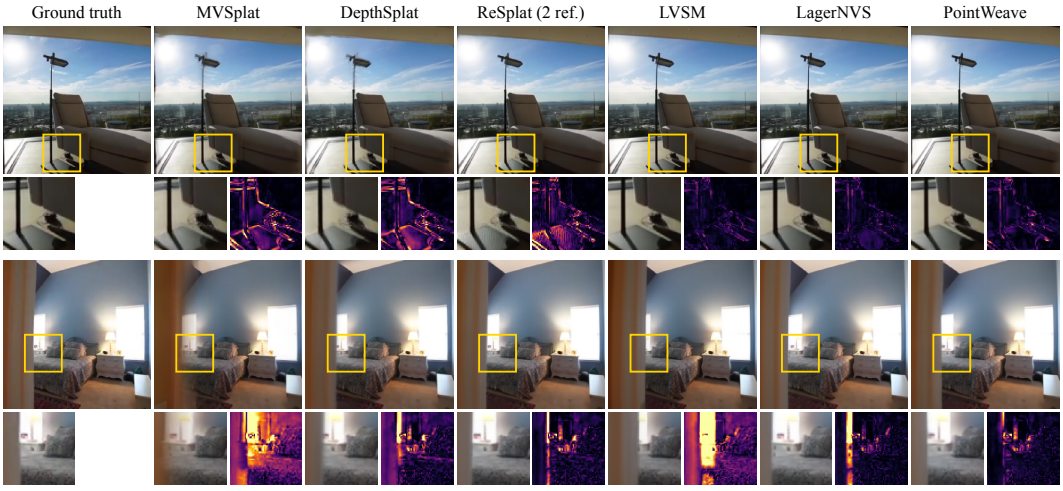

Figure 8: **Additional RE10K examples**, using the layout of Figure 2: the target, a $4\times$ enlarged crop (yellow box) and its error map. Brighter pixels indicate larger errors.

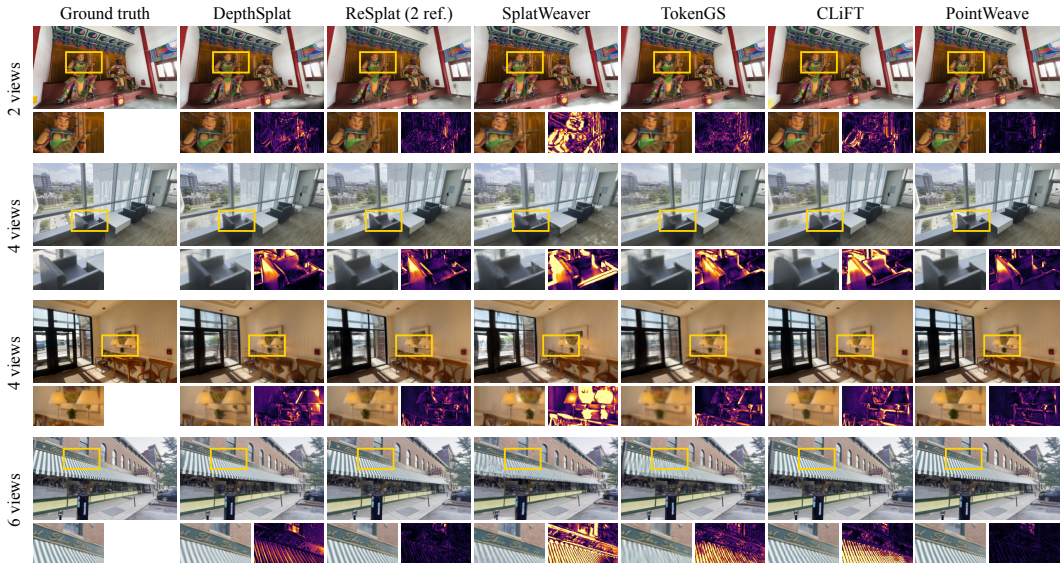

Figure 9: **Additional DL3DV examples with two to six input views**, using the full-frame test and the layout of Figure 2. CLiFT's released model is trained with four and six input views only.

score. Transforming only the means (μ) moves the Gaussians but keeps their shapes fixed. Table 7 uses the test cases, settings and metrics of Table 5.

When the covariances are transformed, ReSplat scores higher than PointWeave under roll, field-of-view and pixel-aspect changes, because its Gaussians move exactly with the camera. When only the means move, ReSplat's elongated Gaussians break down under roll, and all 3DGS-based methods break down under world scaling.

We also fine-tune PointWeave for 500 steps with PVSM's camera augmentation, using 16 scenes per step. With probability 0.5, each target view is rolled by one of PVSM's 18 training angles, rotated by up to ± 25 degrees of yaw and ± 30 degrees of pitch, given a horizontal field of view between 10 and 120 degrees, or given a pixel aspect ratio between 0.8 and 1.25. The loss uses only valid pixels. The fine-tuned model reaches ReSplat's roll and pixel-aspect scores and exceeds its field-of-view score, while its RE10K PSNR changes from 31.38 to 31.13 dB.

Table 7: **Consistency of 3DGS-based methods** on the benchmark of Table 5. Σ: means and covariances transformed; μ: means only. With transformed covariances, the world-scale score equals the untransformed score. + aug.: fine-tuned for 500 steps with camera and world-scale augmentation.

	Roll		FOV		Anisotropy		Scale	
Method	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
MVSplat (Σ)	24.82	.846	19.66	.929	22.21	.908	24.08	.812
MVSplat (μ)	24.62	.844	18.69	.924	21.63	.871	15.73	.622
DepthSplat (Σ)	26.12	.878	20.64	.943	22.70	.913	25.45	.849
DepthSplat (μ)	23.71	.846	19.07	.937	21.87	.878	15.45	.624
ReSplat, no refinement (Σ)	27.64	.894	24.53	.956	26.80	.948	27.07	.870
ReSplat, no refinement (μ)	15.24	.604	18.14	.925	18.35	.840	11.43	.379
ReSplat, 2 refinements (Σ)	27.85	.896	24.70	.957	27.03	.949	27.35	.872
ReSplat, 2 refinements (μ)	15.47	.611	18.20	.925	18.49	.843	11.33	.376
PointWeave	21.85	.790	21.67	.953	24.41	.932	25.20	.805
PointWeave + aug.	27.84	.891	25.42	.957	26.98	.948	26.80	.859