GAUSSIAN FOR GAUSSIAN: UNIFYING FEATURE REPRESENTATION AND RENDERING IN FEED-FORWARD DRIVING SCENE RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vehicle-mounted surround-view cameras have limited overlap, making it difficult to exploit cross-view information while preserving local detail and reconstructing complete novel views. Existing methods address this challenge through cross-view matching or dense spatial grids, but face limited correspondence support or a resolution–memory trade-off. We propose Gaussian for Gaussian, a feed-forward framework that uses cross-camera features as spatially sparse supplements to local features before constructing a separate renderable scene. Using estimated metric depth, we lift multi-scale image features into Feature Gaussians in a shared 3D space. We splat these Gaussians into the source views and fuse the projected features with local image features, retaining pixel-aligned detail alongside geometrically aligned support. To turn the fused features into high-quality novel views, we further predict independent Render Gaussians with layered geometry and learnable local offsets, providing geometric flexibility beyond the initial depth surfaces. Under known ego-motion, our complete system achieves 28.803 dB PSNR on nuScenes, surpassing the DrivingForward baseline by 7.136 dB, and transfers to PandaSet without fine-tuning with a 4.611 dB advantage. Qualitative and regional analyses show improved reconstruction in newly revealed regions and multi-view overlap regions, while feature aggregation yields larger gains where cross-camera support is stronger.

1 INTRODUCTION

Feed-forward approaches to 3D Gaussian Splatting (3DGS; Kerbl et al., 2023) predict renderable scenes directly from images, without per-scene optimization (Charatan et al., 2024; Chen et al., 2024; Szymanowicz et al., 2024). This makes them useful for driving simulation and data augmentation. In driving scenes, however, surround-view cameras face outward to cover different directions, leaving little overlap between adjacent views. On nuScenes, this overlap is only about 10%–15% (Tian et al., 2025; Wei et al., 2025)(Fig. 1). Local image features therefore provide most of the available detail, while other cameras offer complementary observations in limited regions. Novel views must also reconstruct regions newly revealed by vehicle motion. We study novel-view synthesis from a single set of surround-view images under known ego-motion and ask: **How can we better use cross-view information and reconstruct more complete novel views from low-overlap surround-view observations?**

Existing methods differ in how they combine cross-view information and represent the scene. MVSplat (Chen et al., 2024) builds cost volumes by evaluating feature matches across depth candidates. Under limited overlap, this dense search is still performed for many pixels with no corresponding observation in another camera. DrivingForward (Tian et al., 2025) uses scale-aware depth to position Gaussians, but predicts their appearance parameters separately for each image. In our single-frame comparisons, its reconstructions still contain errors in multi-view overlap regions and gaps in newly revealed regions. Omni-Scene (Wei et al., 2025) combines pixel- and volume-based Gaussians to preserve detail and extend coverage. Its triplane features reduce the cost of a dense 3D volume, but increasing the volume branch's spatial coverage or resolution still requires more storage.

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.

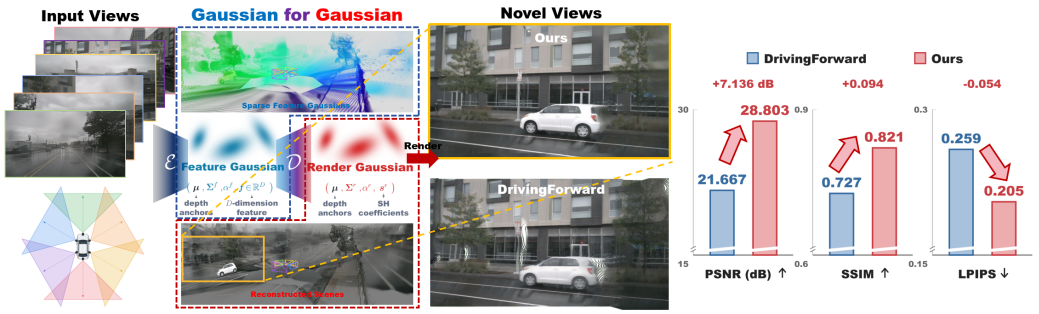

Figure 1: Gaussian for Gaussian improves cross-view information use and novel-view completeness. Given surround-view inputs (left), Feature Gaussians provide sparse cross-camera support for local image features (top). Separate Render Gaussians form the scene for novel-view synthesis (bottom). The image comparison shows more complete newly revealed regions, filling gaps left by DrivingForward; the rightmost charts summarize the complete system's improvements on nuScenes.

These observations motivate three requirements for scene reconstruction from low-overlap surround-view images. First, the intermediate representation should organize multi-camera features in 3D with low memory cost. Second, cross-camera features should provide spatially sparse supplements to local image features, preserving local detail even where views do not overlap. Third, the final scene should have enough geometric flexibility to reconstruct regions newly revealed by vehicle motion, rather than remain limited to the initially visible surfaces.

To meet these requirements, we propose Gaussian for Gaussian, a feed-forward framework with separate Gaussian representations for intermediate features and the final renderable scene. Using estimated metric depth, we lift multi-scale image features into a shared 3D space as Feature Gaussians. This sparse representation stores features at depth-initialized locations rather than throughout a dense scene grid. We then splat the Feature Gaussians from all six cameras into each source view and fuse the projected features with its local image features at each scale. Cross-camera information thus contributes where projection provides support, while local features preserve pixel-aligned detail throughout the image.

To turn the fused features into high-quality novel views, we further predict separate Render Gaussians with layered geometry and learnable local offsets. The depth layers place additional Gaussians behind the first estimated surface, while the offsets allow their positions to adjust for rendering. These choices give the final scene geometric flexibility beyond the initial depth surfaces, supporting the reconstruction of newly revealed regions.

On nuScenes, our complete system achieves 28.803 dB PSNR under known ego-motion, surpassing DrivingForward by 7.136 dB. Without fine-tuning, it also outperforms DrivingForward by 4.611 dB on PandaSet. Qualitative comparisons show more complete newly revealed regions, while regional analysis shows improved reconstruction in multi-view overlap regions. Retrained controls further show that feature aggregation yields larger gains where cross-camera support is stronger. These findings support sparse feature supplementation followed by flexible scene prediction.

Our main contributions are:

- We propose Gaussian for Gaussian, a feed-forward framework for novel-view synthesis from low-overlap surround-view images. It uses separate Gaussian representations for intermediate features and the final renderable scene to improve cross-view information use and novel-view completeness.
- We develop geometry-guided, multi-scale feature fusion in which Feature Gaussians sparsely supplement local image features. A separate decoder then converts the fused features into Render Gaussians with layered geometry and learnable local offsets, providing geometric flexibility for novel-view rendering.
- We demonstrate improved reconstruction quality on nuScenes and zero-shot transfer to PandaSet. Qualitative and regional analyses show more complete newly revealed regions, better reconstruction in multi-view overlap regions, and larger aggregation gains where cross-camera support is stronger.

2 Related Work

Feed-forward novel-view synthesis. Generalizable rendering avoids per-scene optimization by conditioning radiance fields on input images (Yu et al., 2021; Wang et al., 2021). Geometry-aware cost volumes further improve reconstruction by combining evidence from multiple views (Chen et al., 2021; Johari et al., 2022). However, rendering these fields still requires querying many points along each camera ray. Feed-forward 3DGS reduces this cost by predicting Gaussians that are splatted directly into images (Szymanowicz et al., 2024; Charatan et al., 2024; Chen et al., 2024). Further work improves detail through coarse-to-fine refinement (Tang et al., 2025) and represents hidden surfaces with additional depth layers and local offsets (Szymanowicz et al., 2025). We use efficient splatting for both intermediate feature projection and final image rendering. Instead of sampling and interpolating voxel features along camera rays, we splat Feature Gaussians to supplement local image features. Separate Render Gaussians then provide layered geometry and local offsets for novel-view rendering.

Driving-scene reconstruction. Scene-specific driving reconstruction fits static surroundings and moving objects to recorded sequences (Tonderski et al., 2024; Zhou et al., 2024b; Yan et al., 2024). Feed-forward methods avoid this test-time fitting by learning reconstruction priors, including those distilled from offline fields and pretrained geometry models (Wang et al., 2024; Lin et al., 2025). Temporal context further supports dynamic reconstruction (Yu et al., 2026). With only a single surround-view timestamp, however, limited overlap makes it important to preserve local detail while sharing observations and reconstructing newly revealed content. DrivingForward uses scale-aware depth to locate Gaussians but predicts their remaining attributes per image (Tian et al., 2025). Omni-Scene extends coverage through complementary pixel-based and volume-based Gaussians (Wei et al., 2025). Both therefore provide relevant baselines for our low-overlap setting. Unlike DrivingForward and Omni-Scene, we explicitly use sparse Feature Gaussians to carry shared features and fuse these cross-camera features before predicting rendering attributes.

Intermediate 3D representations. BEV representations bring multi-camera image features into a shared frame for perception (Philion & Fidler, 2020; Li et al., 2022). To retain finer 3D structure, occupancy methods use volumes or orthogonal feature planes (Huang et al., 2023; Wei et al., 2023; Zhang et al., 2023). Sparse Gaussians offer an alternative to these grids for representing geometry and semantics (Huang et al., 2024). Feature distillation further shows that Gaussians can carry semantic knowledge from pretrained image models, rather than only color (Zhou et al., 2024a). This ability to carry features also supports neural rendering: feature-splatting methods project learned features into image planes and decode them into RGB (Berriel Martins & Civera, 2024; Li et al., 2024). Rendering in latent space further connects this approach to diffusion, which refines the rendered latents to recover missing image details (Hirschorn et al., 2026). We use both strengths of Gaussians: their ability to carry rich scene information and their efficient splatting. Feature Gaussians aggregate image features for scene prediction, while separate Render Gaussians produce final images directly, without target-view neural decoding or diffusion refinement.

3 Method

3.1 Overview

Given six surround-view images $\{I_i\}_{i=1}^{6}$ at T_0 and known ego-motion to T_1, we synthesize six target images in one forward pass (Fig. 2). Our design follows the three requirements introduced above: a memory-efficient intermediate 3D representation, sparse cross-camera supplementation that preserves local detail, and flexible scene geometry for reconstructing newly revealed regions.

Using metric depth from a frozen network, we lift multi-scale encoder features into a shared 3D space as Feature Gaussians. We splat these Gaussians into each source view to aggregate features across cameras. The projected features are then fused with local encoder features, preserving pixel-aligned detail alongside spatially sparse cross-camera support. A separate decoder uses the fused features to predict Render Gaussians with layered geometry and local offsets. We render the resulting scene at the T_1 camera poses determined by the known ego-motion.

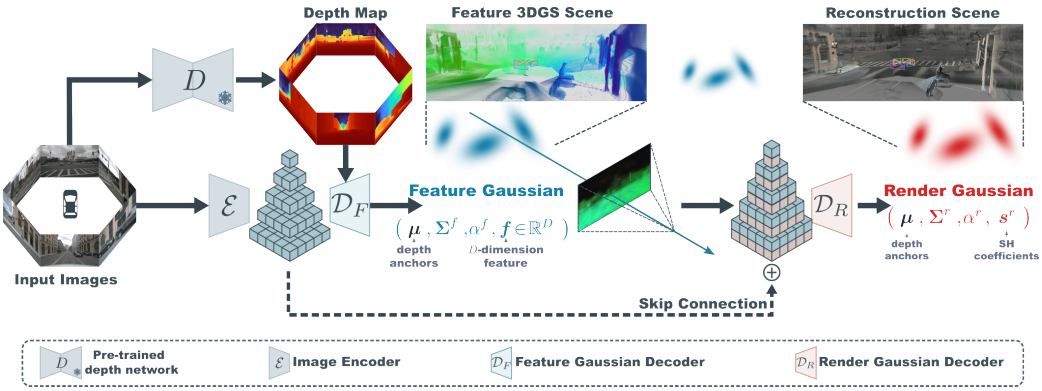

Figure 2: Sparse feature support followed by flexible scene prediction. Six T_0 images yield five-scale features and frozen UniDepthV2 depth. A depth decoder takes the original pyramid and predicts a positive increment; both Gaussian stages share the layered depths. Feature Gaussians splat geometrically aligned support into each T_0 view, where fusion retains local image features. Render Gaussians use layered geometry and local offsets to render T_0/T_1 under known calibration and ego-motion.

The two representations differ in what they carry:

$$g^F = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \alpha, \boldsymbol{f}), \qquad g^R = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \alpha, \boldsymbol{s}), \tag{1}$$

where $\boldsymbol{f}$ is an image feature and $\boldsymbol{s}$ denotes spherical-harmonic appearance coefficients. Both use positions $\boldsymbol{\mu}$, covariances $\boldsymbol{\Sigma}$, and opacities α, but predict their parameters separately. They share initial depths; Render Gaussians additionally learn local position adjustments.

3.2 Image Encoding and Feature-Gaussian Construction

To align image features through geometry despite limited overlap, we associate them with depth-positioned Gaussians. The image encoder $\mathcal{E}$, a ResNet-50 (He et al., 2016), extracts a five-level pyramid $\mathbf{F}_i$ from each image I_i using RGB, metric depth, and ray embeddings. A frozen UniDepthV2 ViT-L/14 network $\mathcal{N}_d$ (Piccinelli et al., 2025) estimates $\widetilde{D}_i = \mathcal{N}_d(I_i)$. We calibrate it as $\widehat{D}_i = s\widetilde{D}_i + b$, with learnable scalars s, b shared across views.

For each pixel $\mathbf{p}$, backprojection gives the base Feature-Gaussian center:

$$\boldsymbol{\mu}_i(\mathbf{p}) = \Pi^{-1}\left(\mathbf{p}, \widehat{D}_i(\mathbf{p})\right), \tag{2}$$

where Π^{-1} uses the corresponding camera intrinsics to lift the pixel into source-camera coordinates. A separate depth decoder takes the five-level source-image pyramid $\mathbf{F}_i$, before aggregation, and predicts a positive depth increment. Adding it to $\widehat{D}_i$ gives the second depth layer. Both Gaussian stages use these two layers to initialize their positions via Eq. 2.

The Feature-Gaussian decoder $\mathcal{D}_F$ predicts opacity, scale, and rotation; scale and rotation determine covariance. Each primitive carries the encoder feature from the corresponding pyramid level, rather than a predicted color or appearance vector. Known calibration and poses transform the six Gaussian sets into a shared T_0 reference frame for aggregation.

3.3 Cross-Camera Feature Splatting

We aggregate features independently at the five scales, omitting the scale index below. For receiving camera j, Feature Gaussians from all six cameras are jointly rasterized into its T_0 plane. Frustum culling and projected footprints select the primitives that contribute to each pixel, making cross-camera support spatially localized. The aggregated feature at pixel $\mathbf{p}$ is

$$\mathbf{A}_j(\mathbf{p}) = \sum_{k \in \mathcal{N}_j(\mathbf{p})} T_k(\mathbf{p}) \alpha_k(\mathbf{p}) \boldsymbol{f}_k, \qquad T_k(\mathbf{p}) = \prod_{m<k} \left(1 - \alpha_m(\mathbf{p})\right), \tag{3}$$

where $\mathcal{N}_j(\mathbf{p})$ contains the depth-sorted Gaussians covering pixel $\mathbf{p}$. The response $\alpha_k(\mathbf{p})$ combines primitive opacity with its projected 2D footprint, and T_k is the accumulated transmittance. We use

standard 3DGS alpha compositing (Kerbl et al., 2023) to combine image features rather than colors. Features from the receiving camera and the other five cameras are composited together, with a zero feature background.

This projection differs from the ray-sampled voxel implementation used in our comparison: that implementation interpolates voxel features at several depths along each camera ray and combines the samples into image-plane features. Feature Gaussians instead splat their image features directly from their 3D positions, without this intermediate ray-sampling step.

Projection supplies information only where Gaussians contribute, whereas local encoder features remain available throughout the source image. We therefore retain both rather than replacing local features with the projected map:

$$\mathbf{H}_j = \phi([\mathbf{F}_j, \mathbf{A}_j]), \tag{4}$$

where $[\cdot,\cdot]$ denotes channel concatenation. Each scale has its own lightweight fusion block ϕ, and the five fused feature maps form the input pyramid for Render-Gaussian prediction.

3.4 Render Gaussians with Layered Geometry

The fused pyramid supplies image information; a separate scene representation determines how that information is rendered after viewpoint changes. The Render-Gaussian decoder $\mathcal{D}_R$ predicts two Gaussians per source pixel from $\mathbf{H}_j$, each with independent opacity, covariance, spherical-harmonic appearance, and local offsets. Their layered depths and position adjustments provide freedom beyond a single depth-initialized surface.

The shared depth decoder in Section 3.2 gives $d_0 = \widehat{D}$ and $d_1 = \widehat{D}+\delta$, with $\delta > 0$. Both Gaussian stages use these depths; Render Gaussians additionally learn local 3D offsets:

$$\boldsymbol{\mu}_g^R(\mathbf{p}) = \Pi^{-1}(\mathbf{p}, d_g(\mathbf{p})) + \Delta\boldsymbol{\mu}_g(\mathbf{p}), \quad g \in \{0,1\}. \tag{5}$$

The offset is predicted in source-camera coordinates and scaled by the corresponding depth, with gradients stopped through this scale factor. Layered depths place primitives beyond the first estimated surface, while offsets let them move away from their initial positions to accommodate imperfect depth and newly visible regions.

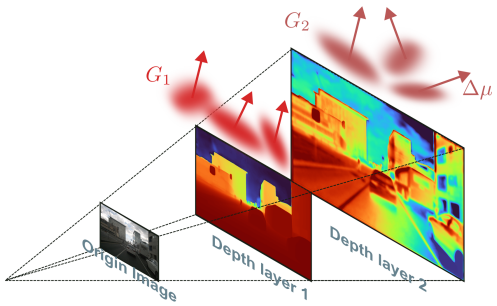

Figure 3: Layered Render Gaussians. Two depth layers initialize the primitives associated with each source pixel. Learned offsets adjust their positions for target-view rendering. Arrows illustrate these adjustments; offsets are predicted for both layers.

We transform the six Render-Gaussian sets into the shared reference frame and render T_0/T_1 using standard Gaussian splatting (Kerbl et al., 2023) and known poses. Further target poses reuse this scene without neural decoding. The T_1 images provide supervision only, not network inputs.

3.5 Training Objective

We supervise both T_0 reconstruction and T_1 novel-view synthesis. The training objective, averaged over valid training views, is

$$\mathcal{L} = \mathcal{L}_1 + \lambda_2 \mathcal{L}_{\text{DSSIM}} + \lambda_p \mathcal{L}_{\text{LPIPS}}, \tag{6}$$

Here $\mathcal{L}_1$ is the mean absolute RGB error over pixels and channels, averaged over supervised views. We use $\mathcal{L}_{\text{DSSIM}} = (1-\text{SSIM})/2$ (Wang et al., 2004) with $\lambda_2 = 0.85$, and VGG-based LPIPS (Zhang et al., 2018). We set $\lambda_p = 0$ for the first 10k steps and $\lambda_p = 0.01$ afterward. The final configuration uses no additional depth supervision, offset regularization, or Gaussian regularization.

4 Experiments

Our experiments examine how Gaussian for Gaussian improves cross-view information use and novel-view completeness in low-overlap driving scenes. We first evaluate the complete system,

then study how Feature Gaussians supplement local features and how Render Gaussians provide geometric flexibility for reconstructing newly revealed regions. Regional analysis identifies where these gains occur. Finally, we compare the reconstruction quality and resource use of Gaussian and voxel intermediate representations.

4.1 Experimental Setup

We follow DrivingForward's single-frame evaluation protocol (Tian et al., 2025) on nuScenes (Caesar et al., 2020), synthesizing six novel views at T_1 from six surround-view images at T_0. The complete single-frame test set contains 150 sample tokens and 900 target images. Ours and DrivingForward with official weights use the same frames, ground-truth calibration and ego-motion, and identical full-image PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) implementations. Each retains its own depth prior, so this comparison evaluates the complete systems under known motion.

The main model is trained on the full split at 352×640. To reduce the cost of retraining multiple architectural variants, we conduct the architecture studies on a 2,535-sample training subset at 128×384. The encoder, depth prior, and training budget are held fixed within each comparison. Both settings use the complete single-frame test set. Their reference models reach 28.803 and 28.334 dB PSNR, respectively. Training uses Adam with an initial learning rate of 10^{-4} for 30 epochs; the main checkpoint is the periodic checkpoint retained at step 149,000, not one selected by test performance.

4.2 Comparison with Existing Methods

We first ask whether the complete design improves novel-view quality. Table 1 evaluates Ours and the official DrivingForward weights at 352×640 with the same evaluator. Our complete model achieves 28.803/0.821/0.205 in PSNR/SSIM/LPIPS, improving PSNR by 7.136 dB. Both methods retain their own depth priors and Gaussian capacities: Ours uses two layers, and official DrivingForward uses one. With fixed weights, replacing DrivingForward's depth with UniDepthV2 at inference does not improve PSNR (Appendix B.1). The retrained controls in Section 4.3 separately analyze feature aggregation, geometry and Gaussian capacities. DrivingForward matches its published 21.67/0.727/0.259 at the reported precision. The other published results are taken from DrivingForward (Tian et al., 2025), with evaluation scopes noted in the table.

Table 1: Novel-view synthesis and zero-shot PandaSet transfer. The bottom two nuScenes rows use DrivingForward's complete 150-sample test set. All PandaSet results use the same 408 six-camera pairs, 2448 target images at 352×640, recorded poses, and metrics.

Method	nuScenes			PandaSet (zero-shot)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelSplat (Charatan et al., 2024)*	17.54	0.502	0.519	20.973	0.613	0.437
MVSplat (Chen et al., 2024)*	17.57	0.528	0.477	18.782	0.535	0.497
3DGS (Kerbl et al., 2023)†	19.57	0.599	0.465		N/A	
Omni-Scene (Wei et al., 2025)‡	18.529	0.451	0.382	20.035	0.580	0.448
DrivingForward (Tian et al., 2025)	21.667	0.727	0.259	18.136	0.562	0.417
Ours	**28.803**	**0.821**	**0.205**	**22.747**	**0.671**	**0.290**

nuScenes: 352×640, except ‡Omni-Scene (224×400); †3DGS optimizes three scenes and is not zero-shot. PandaSet: *public RealEstate10K weights, nuScenes weights otherwise. Input adaptations are specified in Appendix F.

For broader context, we evaluate the complete Omni-Scene checkpoint on the same single-frame test samples and six target views. We retain its native 224×400 resolution and regenerated depth priors. This provides a hybrid-representation reference, separate from the matched comparison at 352×640.

Zero-shot transfer to PandaSet. To test the complete system beyond its training scenes and camera rig, we transfer it to PandaSet (Xiao et al., 2021) without fine-tuning. It improves PSNR over DrivingForward by $4.611\,\mathrm{dB}$ (95% sequence-bootstrap CI: $[4.298, 4.941]$; Appendix F), with higher SSIM and lower LPIPS (Table 1).

Figure 4 examines content completeness and local detail. In Sample 09 (top), DrivingForward leaves blank areas in regions revealed by vehicle motion, whereas Ours reconstructs more complete content. In Sample 07 (bottom), Ours preserves vehicle contours and object boundaries more consistently,

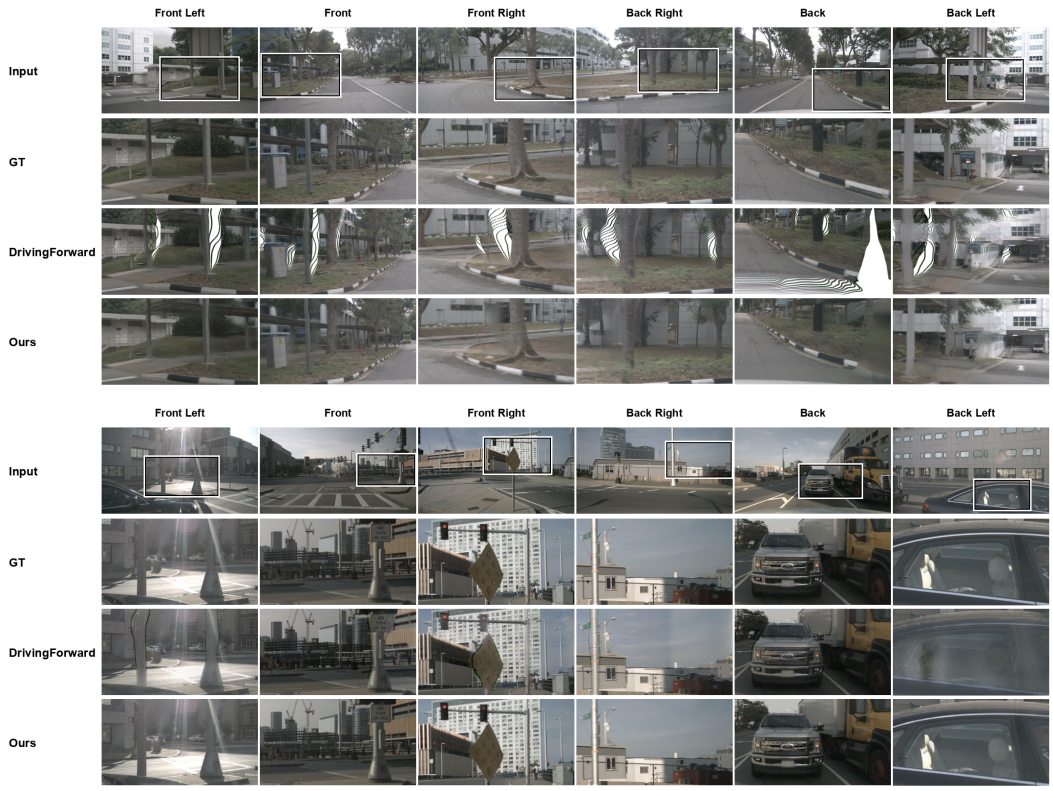

Figure 4: Qualitative comparison with DrivingForward. Top: in Sample 09, Ours reduces exposed white backgrounds and missing content in newly revealed regions. **Bottom:** in Sample 07, Ours reconstructs object boundaries and vehicle details more consistently, with more uniform sky appearance across cameras.

with more uniform sky appearance across adjacent cameras. These examples show the complete system's improvements in both missing content and observed detail; the following controls examine the contributions of feature aggregation and scene geometry.

4.3 Ablation Studies

Table 2 tests the two parts of our design: supplementing local features and giving the final scene geometric freedom. We ask four questions: whether aggregation adds useful information; whether to decode projected features directly or predict Render Gaussians; whether additional depth layers improve reconstruction; and whether local offsets help. Each group compares a retrained variant with its full-model reference, rather than forming a sequence of cumulative additions.

Feature-Gaussian aggregation. The local-only variant zeros $\mathbf{A}$ in $\mathbf{H} = \phi([\mathbf{F}, \mathbf{A}])$ and is retrained with the same fusion interface, decoder, depth prior, and budget. Aggregation raises PSNR from 24.716 to 28.334 and reduces LPIPS from 0.226 to 0.150 (Table 2a), supplementing local-feature reconstruction. Its spatial contribution matters: 70.51% of target pixels receive no other-camera projection support in our analysis (Table 3b). Since $\mathbf{A}$ includes both same-camera and other-camera projections, Section 4.4 examines where the added information improves rendering rather than treating the full-image average as a uniform effect.

Render Gaussians versus direct RGB decoding. Once features can move between views, we ask whether they should directly generate target pixels or first define a renderable scene. Table 2b evaluates both routes on the same T_1 targets. A planar RGB decoder outputs pixels on its input grid, so its features must first be splatted into T_1; decoding a T_0 grid would not generate the target view. Our route instead fuses features at T_0 and predicts Render Gaussians to render T_1, improving PSNR from 27.204 to 28.334 and reducing LPIPS from 0.187 to 0.150. It also separates scene prediction from target-view rendering: another pose requires only RGB splatting of the existing scene, whereas

the direct route repeats feature projection and image decoding. The tested Render-Gaussian route therefore combines higher image quality with a scene reusable across target views.

Layered Gaussian geometry. Feature aggregation supplies information, but the scene must have enough capacity to represent it. Increasing both Feature- and Render-Gaussian layers from $G = 1$ to $G = 2$ raises PSNR from 26.569 to 28.334 and SSIM from 0.822 to 0.839 (Table 2c). Multiple depth hypotheses thus improve on a single layer in this setting. A two-layer DrivingForward variant, fine-tuned from official weights, also improves PSNR from 21.654 to 23.529 in an earlier evaluation; its added fine-tuning is detailed in Appendix C. These results support layered geometry as a complement to feature aggregation.

Table 2: Main ablations at 128×384**.** Each group uses the same training budget. F denotes local image features and A aggregated features. Both routes in (b) synthesize T_1: target-plane RGB decoding versus source-view fusion followed by Gaussian rendering. Group (c) changes both Feature- and Render-Gaussian layers.

Variant	PSNR↑	SSIM↑	LPIPS↓
(a) Feature-Gaussian aggregation			
Local image features only: $\phi([F, 0])$	24.716	0.778	0.226
Local + aggregated features: $\phi([F, A])$	**28.334**	**0.839**	**0.150**
(b) Final image synthesis			
Direct target-view RGB decoding	27.204	0.823	0.187
Render-Gaussian prediction and rendering	**28.334**	**0.839**	**0.150**
(c) Gaussian layers			
$G = 1$	26.569	0.822	0.175
$G = 2$	**28.334**	**0.839**	**0.150**
(d) Render-Gaussian local offsets			
Without offsets	27.937	0.838	0.161
With offsets	**28.334**	**0.839**	**0.150**

Learned local offsets. Adding a second depth layer increases capacity, but without local offsets, Gaussian centers remain constrained to the corresponding source-camera rays. Local offsets allow Gaussians in both layers to adjust their positions in 3D rather than only along these rays, providing greater flexibility for reconstructing newly revealed regions. Removing offsets and retraining reduces PSNR from 28.334 to 27.937 and increases LPIPS from 0.150 to 0.161 (Table 2d). This result supports allowing primitive positions to adjust beyond their ray-aligned initialization.

4.4 Cross-View Support and Novel-View Completeness

The full-image controls establish gains from aggregation and geometry, but do not show where those gains occur. Table 3 examines the two aspects of our question: content recovery under weak projection coverage and reconstruction where views overlap. We then isolate aggregation gains by cross-camera Feature-Gaussian support strength. All comparisons retain the same six input cameras.

Reconstruction under different projection coverage. Using our model's predicted Gaussian centers, we group target pixels by source-camera projection count: Zero, Single, or Multi. Measured LiDAR depth provides a geometric reference to check and correct labels wherever it determines a unique class; predicted labels remain elsewhere. The three regions cover 100% of pixels, and both methods use identical masks. Appendix C details this correction.

Table 3: Regional reconstruction quality and Feature-Gaussian gains. (a) Image-region mean PSNR at 352×640 on LiDAR-checked predicted coverage. Zero/Single/Multi denote projections from 0/1/2+ source cameras and sum to 100%. (b) Pooled-error PSNR at 128×384, grouped by other-camera Feature-Gaussian support, measured by accumulated projected opacity. From Q1 to Q4, support strengthens and the gain over F-only increases.

(a) LiDAR-checked coverage

Region	%	Ours	DF	Δ
Zero	1.70	27.349	15.182	+12.167
Single	78.68	28.911	23.831	+5.080
Multi	19.62	29.524	20.805	+8.719

(b) Other-camera support

Region	%	Full	F-only	Δ
None	70.51	25.755	23.954	+1.801
Q1	7.37	25.217	23.488	+1.729
Q2	7.38	25.745	23.852	+1.892
Q3	7.38	25.253	23.315	+1.938
Q4	7.36	24.072	21.766	**+2.306**

DF: DrivingForward. %: share of all target pixels. Δ: PSNR difference in dB. All methods use six input views.

Table 3a shows benefits at both ends of the coverage range. In Zero regions, our model improves PSNR by 12.167 dB, recovering content where the projection proxy provides no source support. From Single to Multi, our PSNR rises from 28.911 to 29.524 dB, while DrivingForward falls from 23.831 to 20.805 dB, suggesting more effective cross-camera fusion in overlapping regions. The larger advantage in Multi than Single regions persists across six coverage definitions and both PSNR aggregation schemes (Appendix C.3). Figure 5A shows matched regions and reconstruction crops.

How Feature-Gaussian support improves rendering. We next test the principle that cross-camera features should supplement local features where projection provides support. For this analysis, we splat the full model's T_0 Feature Gaussians into T_1 and measure other-camera support by their accumulated projected opacity at the finest feature scale, upsampled to image resolution. We separate unsupported pixels and divide the rest into four equal-sized groups, Q1–Q4, from weakest to strongest support. Within these fixed regions, the full model's PSNR gain over the retrained F-only variant increases from +1.729 dB in Q1 to +1.892, +1.938, and +2.306 dB in Q2–Q4 (Table 3b). Aggregation helps most where other views provide stronger support. Q4 is also the hardest group for local-only reconstruction, at 21.766 dB: cross-camera information complements local features precisely where they struggle. Figure 5B shows the support groups and matched F-only and full-model crops.

4.5 Intermediate Representation and Efficiency

Having established the value of feature supplementation, we compare Gaussian and voxel implementations under matched training and evaluation settings. Both use multi-scale input: the voxel route resizes encoder levels 3–5 to level 3, concatenates and fuses them with a 1×1 convolution, then constructs the volume. Ray sampling produces one aggregated feature map at level 3. Feature Gaussians instead aggregate separately at all five encoder levels. Both retain local encoder features. This comparison tests pre-fused volumetric features against scale-wise Gaussian aggregation, with nearly identical quality: 28.334 versus 28.330 dB PSNR (Table 4).

Table 4: Intermediate 3D representations: quality and resource use. Quality is evaluated at 128×384. Profiling uses the same A800, batch size 1, and 50 test samples after 10 warm-up samples. Latency measures a CUDA-synchronized model forward pass; memory columns report peak allocated and reserved GPU memory.

Representation	PSNR↑	Latency (ms)	Allocated (MB)	Reserved (MB)
Feature Gaussians	28.334	258.43	**4500.86**	**5180**
Dense voxels	28.330	**252.34**	4829.81	6854

The main difference is memory use. Feature Gaussians reduce peak allocated memory from 4829.81 to 4500.86 MB and peak reserved memory from 6854 to 5180 MB, with similar latency. They thus provide a sparse intermediate representation that preserves reconstruction quality, reuses Gaussian splatting for feature projection, and requires less measured GPU memory in our implementation.

5 Conclusion

Gaussian for Gaussian addresses cross-view information use and novel-view completeness in low-overlap driving scenes. Feature Gaussians supply spatially sparse support to local image features; separate Render Gaussians turn their fusion into a scene with layered geometry and local position freedom. The complete system improves reconstruction on nuScenes and transfers to PandaSet without fine-tuning. Aggregation gains grow where other cameras provide stronger support, while geometry controls show the value of flexible primitive placement. The Gaussian intermediate implementation attains similar quality to dense voxels with lower measured memory. These findings support complementing local features through sparse cross-camera aggregation, then constructing a flexible renderable scene. Future work will address moving objects with temporal context and velocity fields, pose/depth robustness, and broader dataset evaluation.

Limitations. Our joint $G = 1/G = 2$ control does not isolate the second Feature-Gaussian layer's benefit, and errors in its predicted depth may misalign projected features. The method also relies on accurate camera-pose priors.

AI USE STATEMENT

We used generative AI tools to support discussions of methods and experimental results, refine experimental designs, and assist with code implementation. The tools also supported literature search, manuscript drafting and translation, and LaTeX table generation. The authors reviewed and verified the AI-assisted work, made the final research decisions, and take responsibility for the accuracy, originality, and integrity of the manuscript, code, and reported results.

REFERENCES

Tomas Berriel Martins and Javier Civera. Feature splatting for better novel view synthesis with low overlap. In *British Machine Vision Conference*, 2024.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11621–11631, 2020.

David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14124–14133, 2021.

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 *European Conference on Computer Vision*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Or Hirschorn, Omer Sela, Inbar Huberman-Spiegelglas, Netalee Efrat, Eli Alshan, Ianir Ideses, Frederic Devernay, Yochai Zvik, and Lior Fritz. Splatent: Splatting diffusion latents for novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8319–8330, 2026.

Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9223–9232, 2023.

Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. GaussianFormer: Scene as gaussians for vision-based 3d semantic occupancy prediction. In *European Conference on Computer Vision*, 2024.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. GeoNeRF: Generalizing NeRF with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18365–18375, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Jason Ku, Ali Harakeh, and Steven L. Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *Conference on Computer and Robot Vision*, pp. 16–22. IEEE, 2018.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8508–8520, 2024.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Jifeng Lu, Yu Qiao, and Jifeng Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision*, pp. 1–18, 2022.

Junhong Lin, Kangli Wang, Shunzhou Wang, Songlin Fan, Ge Li, and Wei Gao. VGD: Visual geometry gaussian splatting for feed-forward surround-view driving reconstruction. *arXiv preprint arXiv:2510.19578*, 2025. URL https://arxiv.org/abs/2510.19578.

Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pp. 194–210, 2020.

Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. UniDepthV2: Universal monocular metric depth estimation made simpler. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. doi: 10.1109/TPAMI.2025.3628473.

Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3D: Feed-forward generalisable 3d scene reconstruction from a single image. In *International Conference on 3D Vision*, 2025.

Shengji Tang, Weicai Ye, Peng Ye, Weihao Lin, Yang Zhou, Tao Chen, and Wanli Ouyang. HiSplat: Hierarchical 3d gaussian splatting for generalizable sparse-view reconstruction. In *International Conference on Learning Representations*, 2025.

Qijian Tian, Xin Tan, Yuan Xie, and Lizhuang Ma. DrivingForward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7374–7382, 2025.

Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. NeuRAD: Neural rendering for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14895–14904, 2024.

Letian Wang, Seung Wook Kim, Jiawei Yang, Cunjun Yu, Boris Ivanovic, Steven Waslander, Yue Wang, Sanja Fidler, Marco Pavone, and Peter Karkus. DistillNeRF: Perceiving 3d scenes from single-glance images by distilling neural fields and foundation model features. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2021.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Dongxu Wei, Zhiqi Li, and Peidong Liu. Omni-Scene: Omni-gaussian representation for ego-centric sparse-view scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22317–22327, 2025.

Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. SurroundOcc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21729–21740, 2023.

Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. PandaSet: Advanced sensor suite dataset for autonomous driving. In *IEEE International Intelligent Transportation Systems Conference*, 2021. URL https://arxiv.org/abs/2112.12610.

Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In *European Conference on Computer Vision*, pp. 156–173, 2024.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

Haibao Yu, Kuntao Xiao, Jiahang Wang, Ruiyang Hao, Yuxin Huang, Guoran Hu, Haifang Qin, Bowen Jing, Yuntian Bo, and Ping Luo. ReconDrive: Fast feed-forward 4D gaussian splatting for autonomous driving scene reconstruction. *arXiv preprint arXiv:2603.07552*, 2026. URL https://arxiv.org/abs/2603.07552.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Yunpeng Zhang, Zheng Zhu, and Dalong Du. OccFormer: Dual-path transformer for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9433–9443, 2023.

Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21676–21685, 2024a.

Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. DrivingGaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21634–21643, 2024b.

APPENDIX

This appendix documents the implementation and evidence behind sparse feature supplementation and flexible scene prediction. It provides network details, depth and protocol checks, and regional visualizations, followed by additional nuScenes examples, zero-shot PandaSet evaluation, and qualitative transfer to a new camera rig.

A IMPLEMENTATION DETAILS

This section specifies the final model evaluated at 352×640. We also give the feature dimensions and primitive counts at 128×384, the resolution used for internal ablations; optimization and optional modules need not be identical across those controls.

A.1 ARCHITECTURE AND PRIMITIVE ALLOCATION

Image features. We initialize the ResNet-50 body (He et al., 2016) with torchvision ImageNet-1K V2 weights. Its original input convolution is replaced by a newly initialized 10-channel convolution receiving RGB, calibrated metric depth divided by 20, and a six-channel Plücker/ray embedding. Table 5 lists the five encoder levels. The finest feature is taken before batch normalization. The final model uses neither adjacency attention nor depth-guided fusion, and has no conventional FPN neck.

Table 5: Encoder features and nominal Feature-Gaussian counts. The last column counts both layers across all six source cameras at 352×640. Spatial dimensions are height $\times$ width.

Level	Channels	Grid at 352×640	Grid at 128×384	Feature Gaussians
1	64	176×320	64×192	675,840
2	256	88×160	32×96	168,960
3	512	44×80	16×48	42,240
4	1,024	22×40	8×24	10,560
5	2,048	11×20	4×12	2,640

Feature routing and decoding. Separate 1×1 heads predict Feature-Gaussian opacity, anisotropic scale, and quaternion rotation from the coarsest encoder feature. These parameter maps are interpolated to each feature resolution; opacity uses a sigmoid, scale an exponential, and rotation quaternion normalization. Each layer carries the matching encoder feature vector, not RGB or spherical-harmonic (SH) coefficients. The six camera sets are merged in the reference-ego frame and splatted into each *source camera plane at* T_0. At each level, a 1×1 convolution, GroupNorm, and SiLU fuse the concatenated local and aggregated maps back to the encoder width. Two independent ResNet decoders consume the fused pyramid, one for each Render-Gaussian layer. No target RGB or target encoder skip is supplied. Each Render Gaussian has sigmoid opacity, scale $0.01 \exp(\cdot)$, a normalized quaternion, and degree-1 SH appearance: three DC values and nine first-order coefficients across the RGB channels.

Counts and rasterization. With six cameras and $G = 2$ layers, the nominal allocations are

$$N_F^l = 6GH_lW_l, \qquad N_F = \sum_{l=1}^{5} N_F^l, \qquad N_R = 6GHW. \tag{7}$$

At 352×640, these give 900,240 Feature Gaussians and 2,703,360 Render Gaussians per six-view input, or 150,040 and 450,560 per camera. At 128×384, the six-camera totals are 196,416 and 589,824, respectively. These are arithmetic counts before visibility culling, not counts of distinct 3D locations or primitives visible in a particular target. Rendering more target views reuses the predicted representation. Feature splatting uses `gsplat` (Ye et al., 2025) version 1.5.3 with `render_mode="RGB"` and `channel_chunk=32`; the latter processes the feature channels in chunks. The rasterizer handles depth ordering and visibility internally. Each scale uses its own grid and scaled intrinsics, a zero feature background, and premultiplied alpha compositing without division by accumulated opacity.

A.2 Depth and Layered Geometry

The UniDepthV2 ViT-L/14 weights (Piccinelli et al., 2025) remain frozen. Its scalar metric Z depth is calibrated as $\widehat{D} = s\widetilde{D} + b$, with learnable global scalars shared across cameras. The final values are $s = 0.9218$ and $b = -0.0494$. Input RGB is resized/padded to a compatible multiple-of-14 size and depth is mapped back to the requested resolution. Depth provides the Gaussian anchors and an image-encoder input; internal UniDepth feature maps are not passed to the Gaussian network.

A separate top-down depth decoder takes all five source-image encoder levels, before feature aggregation, and predicts the secondary-depth logit r_1. This learned branch is separate from frozen UniDepthV2 and from the Feature- and Render-Gaussian parameter decoders. For a source pixel $\mathbf{p}$, the two depth anchors and Render-Gaussian means in the source-camera frame are

$$\begin{aligned} d_0(\mathbf{p}) &= \widehat{D}(\mathbf{p}), \\ d_1(\mathbf{p}) &= d_0(\mathbf{p}) + \exp(\mathrm{clamp}(r_1(\mathbf{p}), -10, 6)), \\ \boldsymbol{\mu}_g^R(\mathbf{p}) &= \Pi^{-1}(\mathbf{p}, d_g(\mathbf{p})) + \mathrm{stopgrad}(d_g(\mathbf{p}))\, \mathbf{o}_g(\mathbf{p}), \quad g \in \{0, 1\}. \end{aligned} \tag{8}$$

Here Π^{-1} uses the source camera's intrinsics, and $\mathbf{o}_g$ is an unconstrained source-camera XYZ offset. The positive increment is clipped in logit space; the offset has no additional $\tanh$ bound, and only its depth multiplier is detached. Feature and Render Gaussians share depth anchors, not their final positions or covariance parameters. Ground-truth camera calibration and ego poses transform these source-aligned primitives through the T_0 reference-ego frame into the requested target cameras.

A.3 Training Configuration

The final model is trained on 20,096 nuScenes sample tokens for 30 epochs at 352×640, using two GPUs with two six-view samples per GPU. We use Adam with default betas, zero weight decay, and learning rate 10^{-4}, multiplied by 0.1 after global step 60,000; there is no learning-rate warm-up. Training uses FP32 with TF32 enabled, seed 42, and eight data-loader workers per process, without EMA. Evaluation uses the retained checkpoint at step 149,000, not a validation-best selection.

Reconstruction supervision is averaged over valid T_0 and T_1 views. The weights of L_1 and DSSIM are 1 and 0.85, with $\mathrm{DSSIM} = \mathrm{clamp}((1 - \mathrm{SSIM})/2, 0, 1)$. VGG-based LPIPS is enabled only after step 10,000 with weight 0.01. There is no additional depth supervision, Gaussian or offset regularizer, or active velocity-prediction branch in this configuration.

B Depth-Prior Diagnostic

The main comparison uses different depth systems: our model uses frozen UniDepthV2 with learned affine calibration, whereas DrivingForward uses its released depth network. We examine this difference through an inference-time depth substitution and a historical sparse-LiDAR check.

B.1 Inference-Time Depth Substitution

We keep the released DrivingForward SF weights fixed and replace only the metric depth maps supplied to Gaussian backprojection and the Gaussian prediction network. Its original image-feature pathway, $G = 1$ configuration, renderer, and evaluation functions remain unchanged. All variants use the complete DrivingForward single-frame test split: 150 samples and 900 T_1 images at 352×640, with targets defined by `sample_data.next` and recorded camera calibration and ego poses. No retraining or test-time optimization is performed.

We test both raw frozen UniDepthV2 ViT-L/14 depth and the same predictions with our final model's learned affine calibration, held fixed (scale 0.921787, shift -0.049400 m). UniDepth receives only the source RGB images; no target images, LiDAR fitting, or per-image scale alignment are used. The original-depth rerun exactly reproduces the stored baseline's per-sample PSNR, SSIM, and LPIPS.

Direct substitution does not improve PSNR: the raw and calibrated variants reduce it by 0.421 and 0.522 dB, respectively (Table 6). Both slightly improve SSIM, and the calibrated variant also slightly improves LPIPS. Thus, directly replacing DrivingForward's depth with UniDepthV2 does not close

Table 6: DrivingForward with inference-time depth substitution. All rows use the same released SF weights and the complete 150-sample test split at 352×640. Only the input metric depth maps change; the calibrated variant uses the fixed affine parameters from our final model.

Depth source	PSNR↑	SSIM↑	LPIPS↓
Original DrivingForward depth	21.6665	0.7271	0.2589
UniDepthV2 (raw)	21.2456	0.7294	0.2603
UniDepthV2 (our affine calibration)	21.1440	0.7365	0.2576

the PSNR gap. This tests compatibility with fixed reconstruction weights, not a depth-matched retraining.

B.2 Historical Sparse-LiDAR Check

To examine the depth estimates themselves, Table 7 compares the first-layer depth from UniDepthV2 which we used in our model with DrivingForward's official depth predictions on the same first 12 samples of the DrivingForward single-frame test split, comprising 72 camera images at 352×640.

LiDAR points are projected into each camera, and errors are measured at projected valid pixels without spatial dilation (radius zero). Neither median scaling nor per-image scale alignment is applied. We report absolute relative error (AbsRel), root mean squared error (RMSE), mean absolute error (MAE), and δ_1, the fraction of valid pixels satisfying $\max(d/\widehat{d}, \widehat{d}/d) < 1.25$.

Table 7: Existing sparse-LiDAR depth diagnostic. Both rows use the same 12 samples and projection protocol. RMSE and MAE are in meters; δ_1 is a fraction.

Depth prediction	AbsRel↓	RMSE↓	MAE↓	δ_1 ↑
DrivingForward (official depth)	0.1984	7.8980	3.8143	0.7388
Ours (UniDepth-based)	0.3106	8.3198	5.1024	0.1090

DrivingForward has lower depth error in this diagnostic. Thus, the available measurements do not support assuming that the UniDepth-based reconstruction automatically benefits from a more accurate depth estimate. The main comparison evaluates the two complete systems under matched image-evaluation conditions.

C Evaluation and Diagnostic Details

C.1 Test Set and Metric Definitions

We use nuScenes `v1.0-trainval` (Caesar et al., 2020) and DrivingForward's single-frame test split (Tian et al., 2025). Its 150 unique sample tokens come from 150 different scenes. Every sample has six valid T_0 source images and six valid per-camera `sample_data.next` targets, giving 900 evaluated T_1 images. Intrinsics, calibrated-sensor extrinsics, and ego poses are ground-truth nuScenes metadata; target motion is an input, not a prediction.

For the main 352×640 comparison, both methods use the same tokens, target images, and metric definitions, without metric-time cropping, masking, resizing, or border exclusion. PSNR is computed from full-image RGB MSE after clipping prediction and ground truth to $[0, 1]$, then averaged over images. SSIM uses `skimage` with an 11×11 window, Gaussian weights, `channel_axis=0`, and data range 1. LPIPS uses version 0.1.4, the VGG network, and `normalize=True`. Our model renders against a gray background $(0.5, 0.5, 0.5)$.

DrivingForward retains its official SF weights and VFDepth-style depth network; our model uses its separately trained network and frozen UniDepthV2 with learned global affine calibration. Thus evaluation is matched, whereas depth priors, training histories, and training budgets are not matched controls. Omni-Scene uses the same 150 tokens and six T_1 targets through a local adapter, but retains its native 224×400 resolution and separately generated Depth Anything V2-Large/Metric3D-v2-Large priors. Its full pixel- and volume-Gaussian checkpoint is evaluated as a system reference, not a same-resolution carrier replacement.

C.2 Scope of the Mechanism Controls

The 128×384 studies use the 2,535-sample training subset and the complete 150-sample test set. Table 8 states what each retrained comparison changes; the differences are independent comparisons rather than an additive sequence of improvements.

Table 8: Interventions in the mechanism studies.

Control	Intervention and preserved components
Local-only (F)	Set $A = 0$ in $\phi([F, A])$; retain fusion dimensions, decoder, depth prior, $G = 2$, offsets, loss, and training schedule. The removed branch includes both self- and other-camera projections.
Direct RGB	Align features with the evaluated T_1 image plane before FPN RGB decoding; use the projected pyramid and coverage. The main route instead fuses local F in T_0 and predicts Render Gaussians for T_1 rendering.
$G = 1$	Reduce the layer count in both Feature and Render Gaussians, not only the rendering stage.
Zero offset	Disable learned Render-Gaussian offsets and retrain; other reference settings are retained.
Dense voxel	Keep the base model and training budget. Resize and concatenate encoder levels 3–5, fuse them before voxel construction, and project one level-3 map; the Gaussian route aggregates at five scales. Both retain local encoder features.

Additional DrivingForward capacity study. A historical 352×640 experiment initialized a two-layer rendering variant from the official SF weights and fine-tuned it for three epochs on the 2,535-sample subset, with learning rate 2×10^{-5}. The depth, pose, and Gaussian networks remained trainable. The earlier evaluator reported 23.529/0.745/0.236, versus 21.654/0.719/0.279 for the original weights. This tests an expanded and adapted model, whereas our main-table baseline uses the official weights without this fine-tuning and a later unified metric path. The existing retrained Ours $G = 1$ control is at 128×384; it is not a full-data 352×640 capacity-matched main-table model.

Historical direct-feature model. An earlier model decoded the five local encoder levels directly into Render Gaussians, without intermediate 3D feature aggregation or the fusion block. It reached 25.700/0.811/0.189 at 128×384 on the same 150-sample split. This run used velocity prediction and feature-consistency supervision; its LPIPS term had not yet activated. The reference and strict local-only models instead use checkpoints with active LPIPS and no velocity branch. The historical result therefore compares development configurations, whereas Table 2a isolates removal of aggregated features. We report both to distinguish the direct-feature architecture from the matched $\phi([F, 0])$ control.

Table 9: Historical and controlled local-feature models. All use the same 128×384 test split. The first row differs in architecture and losses; only the final pair is a matched aggregation ablation.

Decoder input	PSNR↑	SSIM↑	LPIPS↓
Direct F (historical)	25.700	0.811	0.189
$\phi([F, 0])$ (controlled)	24.716	0.778	0.226
$\phi([F, A])$ (reference)	28.334	0.839	0.150

C.3 Coverage and Other-Camera Support

Predicted coverage with sparse LiDAR checks. For Table 3a, we start from the full model's predicted T_0 Render-Gaussian G_0 centers. We transform them through the reference ego frame into each T_1 camera, retain centers with depth $> 10^{-4}$ and in-image projections, and dilate each source camera's binary mask by radius 2 pixels. The number of source masks covering a pixel defines Zero (0), Single (1), or Multi (≥ 2). These disjoint groups cover every target pixel.

Measured LiDAR depth supplies an independent geometric reference for correcting coverage labels; it does not replace the model's predicted depths. We use the T_0 scan and the nearest T_1 scan within 50 ms, with recorded calibration and ego poses. At target pixels with a LiDAR point, its 3D location is projected into every source camera. A source is visible if its projected depth agrees with

source LiDAR within $\max(0.5\,\mathrm{m}, 0.02d)$ in a radius-1 neighborhood; a nearer source surface marks occlusion. Out-of-image projections are unsupported. Missing or inconsistent neighborhoods remain unresolved. Neighborhoods spanning more than twice the depth tolerance are treated as ambiguous.

Confirmed visible sources give a lower bound on support; adding unresolved sources gives an upper bound. After grouping counts into 0/1/2+, a unique reference exists when these bounds imply the same group. We correct only unique conflicts, at the measured pixel, without propagating corrections. All other predicted labels remain, including pixels without LiDAR and 26,746 partially constrained conflicts. Thus sparse observations never remove pixels from the comparison. This remains a model-dependent projection proxy checked by independent observations, not a dense ground-truth visibility or disocclusion mask.

The checked groups contain 3,440,747, 159,535,238, and 39,776,015 pixels, totaling 202,752,000 pixels in 900 images. Single and Multi are nonempty in every image; Zero is nonempty in 814. Predictions, GT, and masks are joined by sample token, target sample-data token, and camera name. Both methods use identical regions. Main-table PSNR averages nonempty image regions; Table 10 also pools squared error over all pixels before conversion to PSNR.

Table 10: LiDAR-checked region aggregation and uncertainty. All values are in dB. The main table reports image-mean PSNR. Gap confidence intervals use 1,000 paired sample-level bootstrap draws (seed 42), keeping each sample's six cameras together.

Region	Mean gap 95% CI	Pooled Ours	Pooled DF	Pooled gap 95% CI
Zero	[11.465, 12.892]	21.915	6.570	[14.444, 16.441]
Single	[4.753, 5.442]	25.814	21.747	[3.741, 4.403]
Multi	[8.107, 9.387]	26.088	19.914	[5.678, 6.727]

Sensitivity to the LiDAR corrections. Table 11 gives the original predicted-mask results without corrections. Single and Multi change little. Zero's image-mean values change more because pointwise corrections introduce small Zero regions into additional images: 814 have nonempty Zero regions after checking, versus 552 before. These small regions receive the same image-level weight as large regions. Pooled Zero gaps are much less sensitive: 15.370 dB before checking and 15.345 dB after. Both aggregations support improvements in Zero regions and a larger Multi than Single gap, without treating either grouping as ground-truth disocclusion.

Table 11: Original predicted masks before LiDAR corrections. All 150 test samples, matched by camera name. Image-mean averages nonempty regions; pooled PSNR uses dataset-wide regional MSE. Zero, Single, and Multi are disjoint.

	Image-mean PSNR			Pooled PSNR	
Region	Ours	DF	Δ	Ours	DF
Zero	24.457	9.400	+15.057	21.892	6.522
Single	28.911	23.828	+5.083	25.814	21.743
Multi	29.517	20.807	+8.711	26.087	19.913

Figure 5A visualizes the checked masks used in the main comparison. The examples show local content gaps and overlapping-view boundaries; the masks describe projection support, not the cause of every reconstruction error.

Consistency across coverage definitions. We also compare existing partitions constructed from measured LiDAR alone. The depth-consistent partition classifies only target LiDAR points whose source support can be resolved through the depth test above. The frustum partition counts source cameras containing each target LiDAR point, without an occlusion test. For dense partitions, we complete target LiDAR depth with IP-Basic (Ku et al., 2018) or nearest measured depth, then count source-camera frustum projections. Both retain measured depths and classify every pixel; neither completion is dense ground truth.

Table 12 consolidates these alternatives with the predicted and LiDAR-checked masks, using the same cached predictions matched by sample and camera. In every partition, our advantage over DrivingForward is larger in Multi than Single regions, and Zero regions also improve. This holds for both image-mean and pooled PSNR, so the relative pattern is not specific to the primary partition

or averaging convention. Absolute values differ because the masks select different pixels; these are sensitivity checks on the same test set.

Table 12: Coverage-partition sensitivity. Ours minus DrivingForward, in dB, on all 150 test samples. "Pixels" is the fraction classified as Zero, Single, or Multi. Sparse LiDAR partitions evaluate only measured, classifiable points; the other partitions cover the full image.

		Image-mean gap			Pooled gap	
Partition	Pixels (%)	Zero	Single	Multi	Single	Multi
Predicted	100.000	15.057	5.083	8.711	4.071	6.174
LiDAR-checked predicted	100.000	12.167	5.080	8.719	4.067	6.175
Sparse LiDAR, depth-consistent	0.905	10.863	4.490	8.883	3.590	6.940
Sparse LiDAR, frustum	1.511	19.458	4.642	8.727	3.706	6.305
LiDAR + IP-Basic	100.000	16.140	4.973	8.770	3.949	6.369
LiDAR + nearest depth	100.000	16.281	4.939	8.771	3.903	6.317

Feature-Gaussian support. At 128×384, we separately splat the full model's Feature Gaussians from the corresponding T_0 camera and from the other five cameras into each T_1 plane. These target-plane splats are diagnostic only: the main network aggregates features in T_0 planes. We bilinearly upsample other-camera accumulated alpha from the finest feature map (64×192) to RGB resolution, with `align_corners=False`. Pixels with $\alpha_{\text{other}} > 10^{-4}$ are supported. For each sample, supported pixels from all six target cameras are pooled to define four alpha quartiles; the resulting fixed groups are used for both Full and local-only predictions. Unlike the image-region averages above, the reported quartile PSNR pools squared RGB error across all pixels in each group over the dataset before applying $-10 \log_{10}$. This diagnostic measures the model's feature-projection support, not ground-truth visibility or disocclusion.

C.4 Resource Measurement

Multi-scale input to the voxel route. Numbering encoder levels from 1 to 5, the voxel implementation forms

$$F^V = \mathrm{LeakyReLU}\big(\mathrm{Conv}_{1\times1}[F^3, \mathrm{Resize}_3(F^4), \mathrm{Resize}_3(F^5)]\big)\,.$$

The concatenated input has $512+1024+2048 = 3584$ channels and is reduced to 256 at $H/8 \times W/8$. These features populate a $100 \times 100 \times 20$ volume. Projection samples 50 depth bins along each camera ray, then reduces the samples to a 512-channel level-3 map. Other aggregated levels are zero; all five local encoder levels still enter their fusion blocks. The Feature-Gaussian route instead supplies an aggregated map at every level. Thus the comparison tests two implementations of multi-scale intermediate aggregation, with different scale-fusion and projection structures.

The carrier comparison uses one A800, batch size 1, 128×384 images, the same 50 test samples, and 10 warm-up samples. Latency is a CUDA-synchronized model forward pass, not an isolated rasterizer call. Allocated memory denotes memory occupied by tensor allocations; reserved memory includes the CUDA caching allocator's retained pool. Both are peak measurements, not steady-state memory, and the carrier table preserves the recorded MB units. The voxel implementation is slightly faster, while Feature Gaussians use less measured allocated and reserved memory.

Table 13: Additional 352×640 implementation profiles. One A800, batch size 1. Memory is peak allocated GiB. These records have the timing differences described below and are not a standardized speed ranking.

Operation	Method	Time (s)	Memory (GiB)
Six-view inference	Ours	0.305	9.96
Six-view inference	DrivingForward	0.580	4.91
Training step	Ours	0.865	23.69
Training step	DrivingForward	1.479	26.56

Our inference profile times the model call producing six T_1 rendered views (10 warm-up/50 measured samples, TF32 enabled). DrivingForward times batch processing including depth, pose, rendering, and losses (2/5 samples, TF32 disabled). These are six-view measurements; separate per-camera latency was not measured. Training profiles include forward, loss, backward, and optimizer update after 5 warm-up steps over 20 measured steps, using four workers and the same memory-backed

dataset subset. Their token orders and TF32 settings differ. These measurements describe the implemented systems rather than isolate representation cost.

D REGIONAL VISUALIZATIONS

Panel A locates the complete system's improvements in content completeness and overlapping-view reconstruction. Panel B compares the full model with local-only reconstruction to show where Feature-Gaussian support helps. Each row uses identical crops for all predictions and the ground truth.

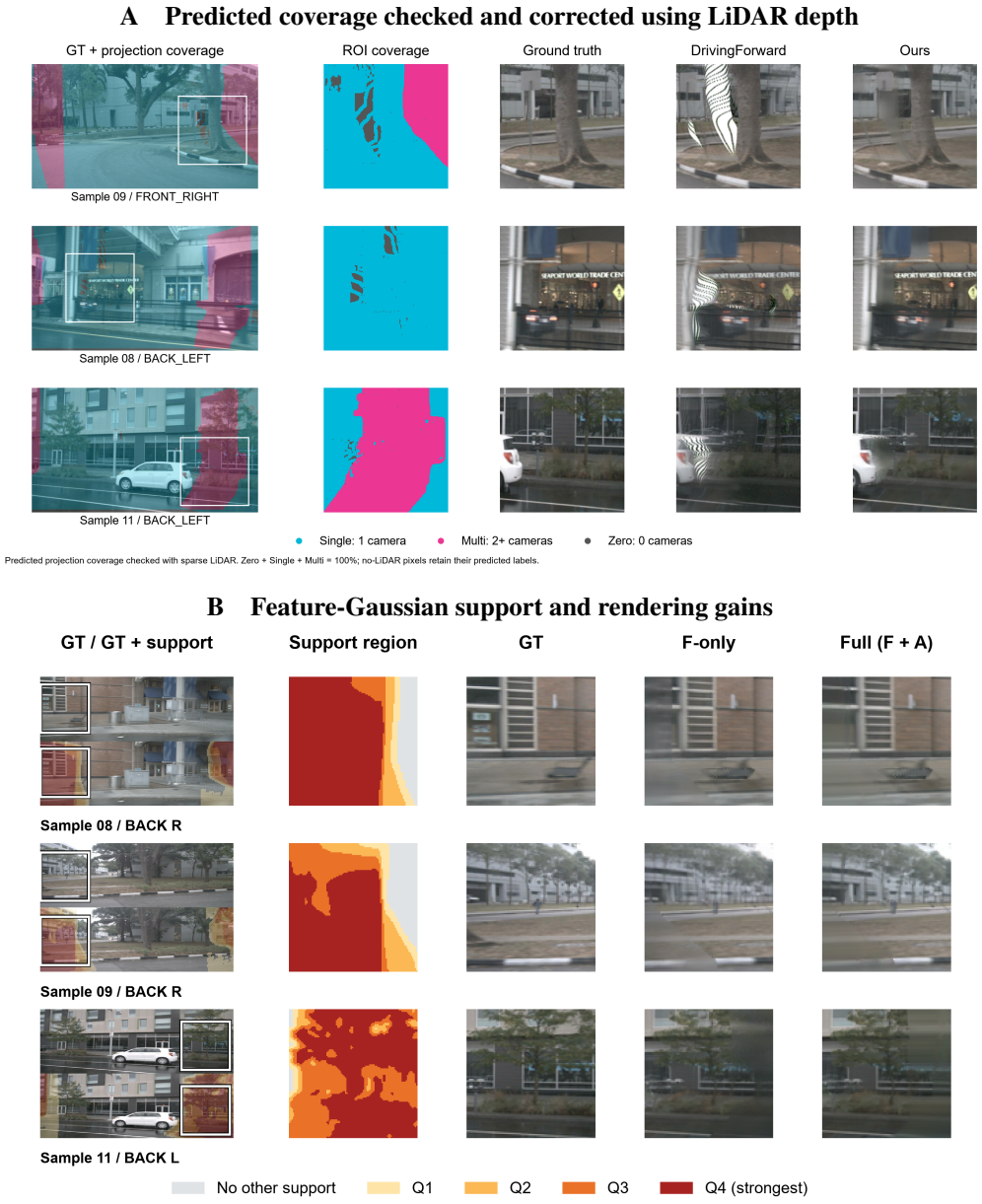

Figure 5: Regional visualizations. A: LiDAR-checked predicted coverage for Table 3a, with aligned GT, DrivingForward, and Ours crops. Gray, cyan, and magenta denote Zero, Single, and Multi; every pixel belongs to one region. The crops show improved content completeness and boundary reconstruction. Samples 09, 08, and 11 retain their original cameras and crops. **B:** Other-camera Feature-Gaussian support and matched GT, F-only, and Full crops for Table 3b. Darker warm colors indicate stronger support; Q4 has the strongest support and largest rendering gain over F-only.

E ADDITIONAL NUSCENES COMPARISONS

Figures 6–8 extend the main-paper comparison to ten additional samples from the DrivingForward test split. The examples are drawn from the existing visualization set selected for visible differences between the methods; quantitative conclusions use the entire test split rather than these selected images. The six camera columns, target frames, and normalized crop coordinates are identical across methods. Each example shows the input images with crop boxes, followed by GT, DrivingForward, and Ours. The later examples include vehicle motion as well as the near-stationary cases in the earlier examples.



Figure 6: Additional six-camera comparisons (1/3). Three samples with identical GT-defined crops across methods. Within each sample, rows are Input, GT, DrivingForward, and Ours; columns follow the camera labels. These selected visual examples supplement, rather than replace, the full-test quantitative comparison.

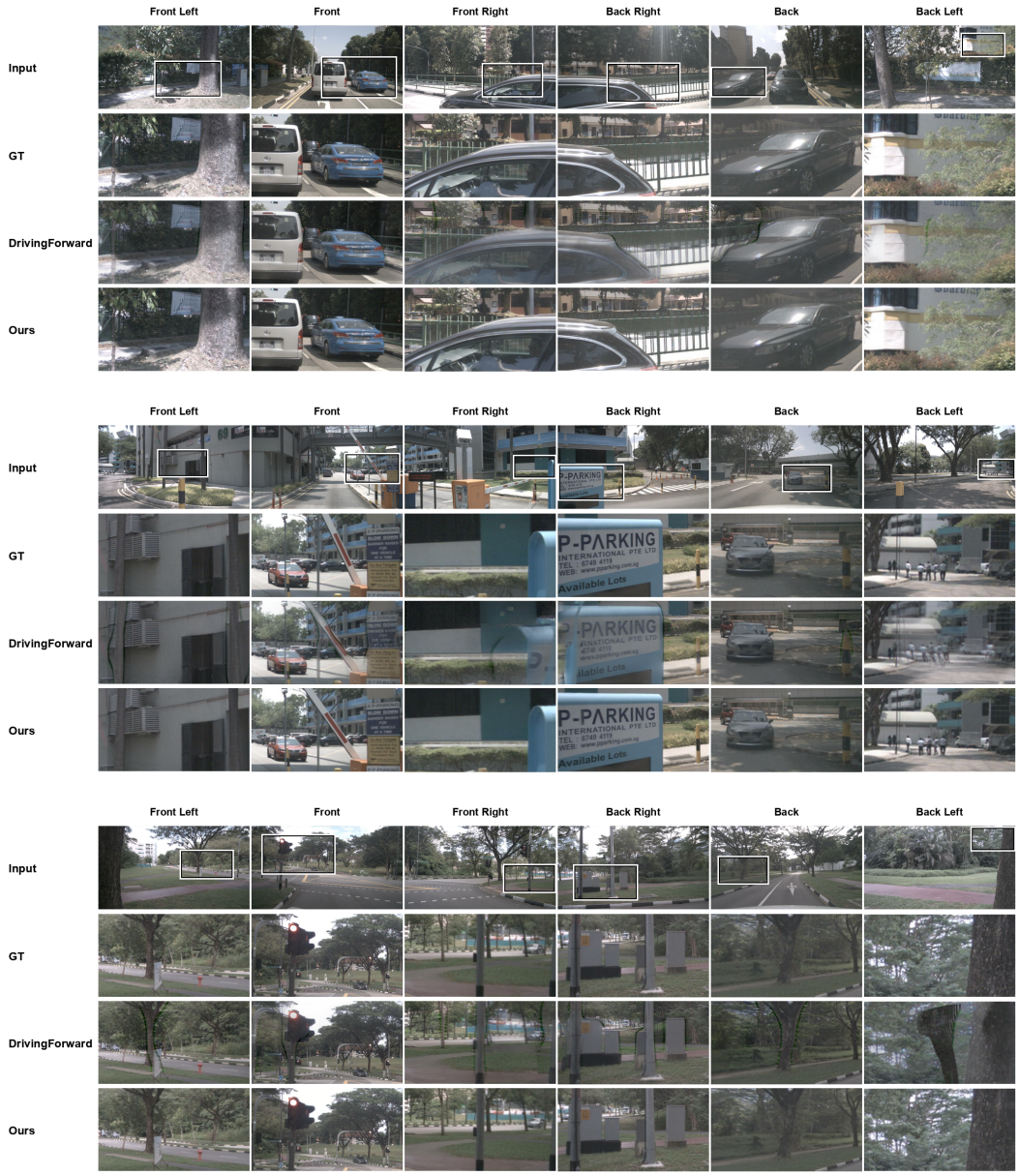

Figure 7: Additional six-camera comparisons (2/3). Three further samples show building surfaces, road boundaries, and foreground objects. The four-row order and shared-crop protocol are unchanged.

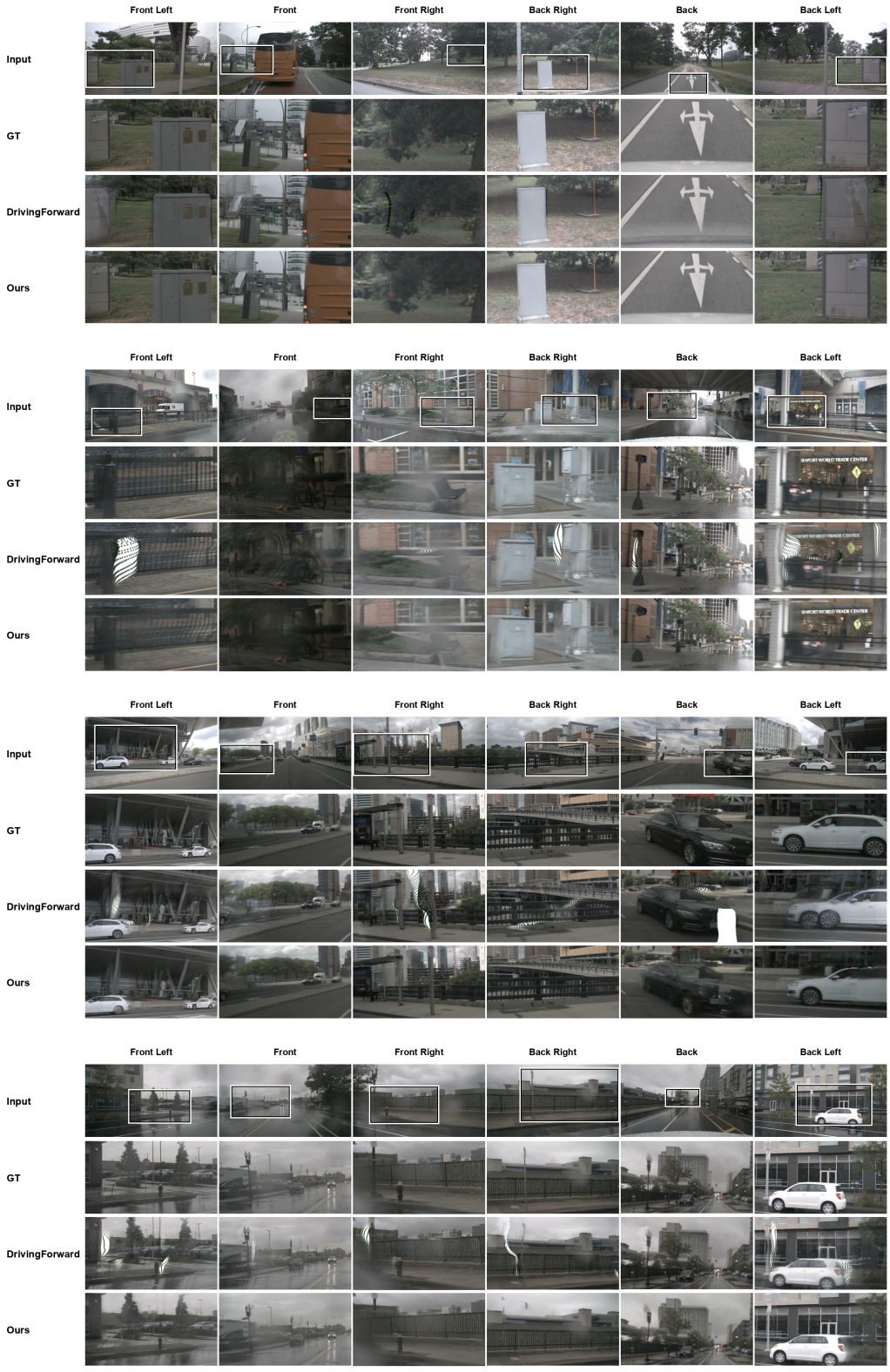

Figure 8: Additional six-camera comparisons (3/3). Four further samples, including motion-containing examples in the last three blocks. Crops are selected on GT and reused unchanged for both predictions.

F ZERO-SHOT GENERALIZATION TO PANDASET

Fixed evaluation set. We evaluate transfer to PandaSet (Xiao et al., 2021) without training or fine-tuning on that dataset. For each available sequence, we select source frames 0, 20, 40, and 60 and use the next frame as the target. All six camera pairs must be valid, with a recorded time interval between 0.05 and 0.20 seconds. Sequence 054 has invalid timestamps and is excluded before inference. This gives 102 sequences, 408 six-camera pairs, and 2448 target images, with an approximately 0.1-second prediction interval. This is our fixed transfer protocol, not an official PandaSet novel-view benchmark.

Common targets and scoring. All methods receive the same source observations and recorded calibration and camera poses. We decode RGB images with PIL and resize full images with Lanczos interpolation, without cropping or color correction. All target images and renderings are 352×640. We verify the identity of target pixels across methods by per-image hashes and compute the same full-image PSNR, Gaussian-window SSIM, and VGG LPIPS as in our matched comparison. Targets are used only for evaluation, not as model inputs or for model selection.

Pretrained models and input adaptation. Ours, DrivingForward, and Omni-Scene retain their nuScenes-trained reconstruction weights. Ours and DrivingForward use 352×640 inputs. Omni-Scene retains its 224×400 feature input and complete pixel-plus-volume model, with source metric depth and confidence generated by the released Metric3D-v2 ViT-Large model. Its predicted Gaussians are rendered directly at the common target resolution, rather than upsampling a low-resolution RGB prediction. Reference-frame axes are matched to each model's training convention; relative camera poses remain the recorded ones.

For pixelSplat and MVSplat, we use the public RealEstate10K checkpoints. Both receive full-view 352×640 images with correspondingly scaled intrinsics. MVSplat processes six source views jointly with its pretrained parameters and released depth bounds. pixelSplat's two-view checkpoint does not contain the additional view embeddings of a multi-view-trained model. We therefore pair each source camera with the camera whose optical axis is closest, apply the unchanged two-view network, and retain only the first camera's Gaussians. Combining these six sets gives one contribution per source camera, without duplicate primitives. Pair selection uses calibration alone. Both use their official Gaussian rasterizer, with the complete recorded target intrinsics, including the principal point.

These are comparisons of available pretrained systems with different training domains and priors. RealEstate10K-to-PandaSet transfer is distinguished from nuScenes-to-PandaSet transfer in Table 1. Original 3DGS requires scene-specific optimization and has no zero-shot checkpoint, so its transfer entry is not applicable. Figures 9 and 10 show two predefined examples from this test set.

Paired uncertainty. For the nuScenes-trained Ours and DrivingForward checkpoints, the mean PSNR difference is $4.611\,\mathrm{dB}$. A paired bootstrap over the 102 sequences, with 2000 resamples, gives a 95% interval of $[4.298, 4.941]\,\mathrm{dB}$. The corresponding intervals are $[0.102, 0.116]$ for the SSIM increase and $[0.120, 0.132]$ for the LPIPS reduction. Resampling sequences rather than individual images retains the grouping of cameras and timestamps within each sequence.

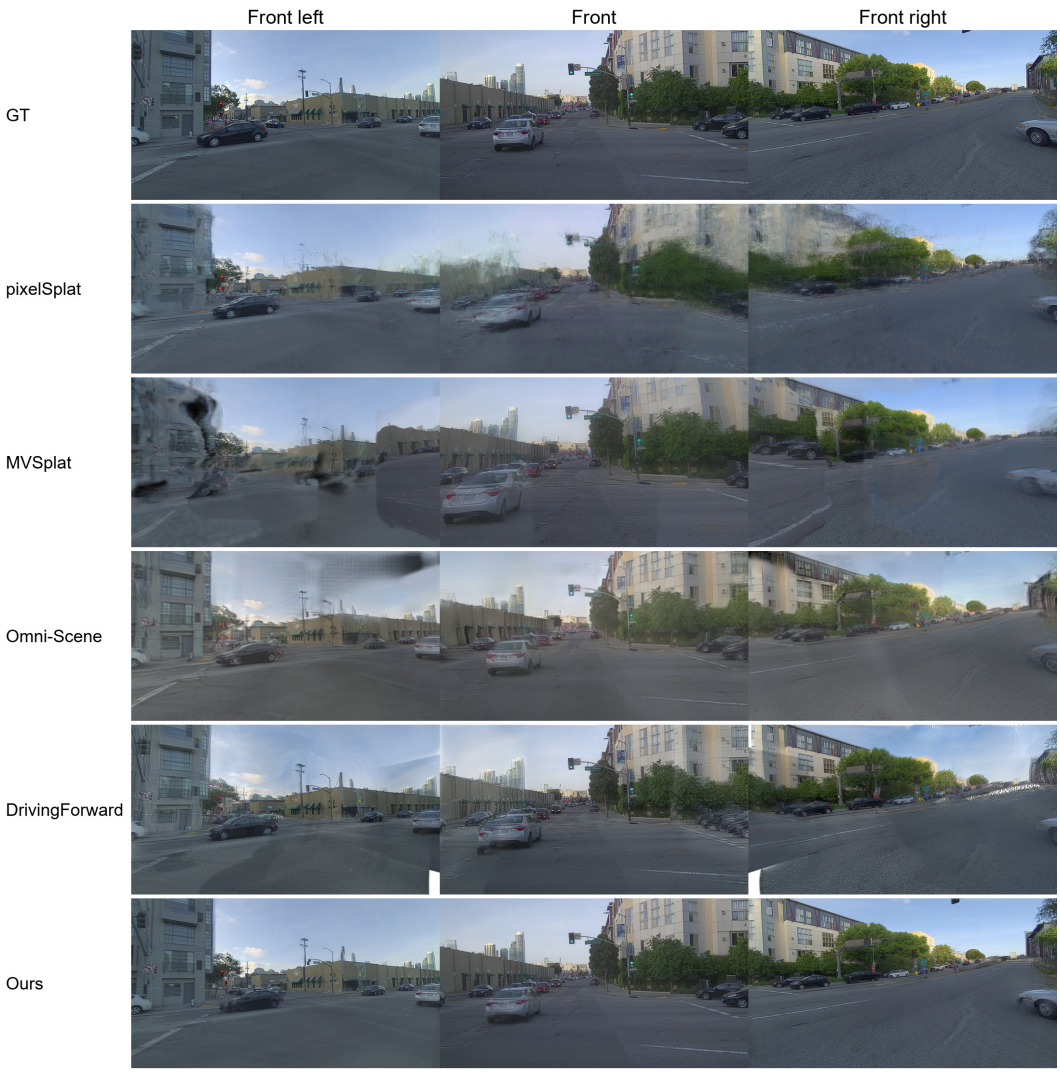

Figure 9: PandaSet zero-shot comparison, sequence 001. Three forward-facing target views at frame 1, reconstructed from all six source views at frame 0. pixelSplat and MVSplat use RealEstate10K weights; the other models use nuScenes reconstruction weights. This is the first predefined visualization sample. All methods use the same target poses and complete 352×640 images, without cropping or color correction. Quantitative evaluation includes all six target cameras.

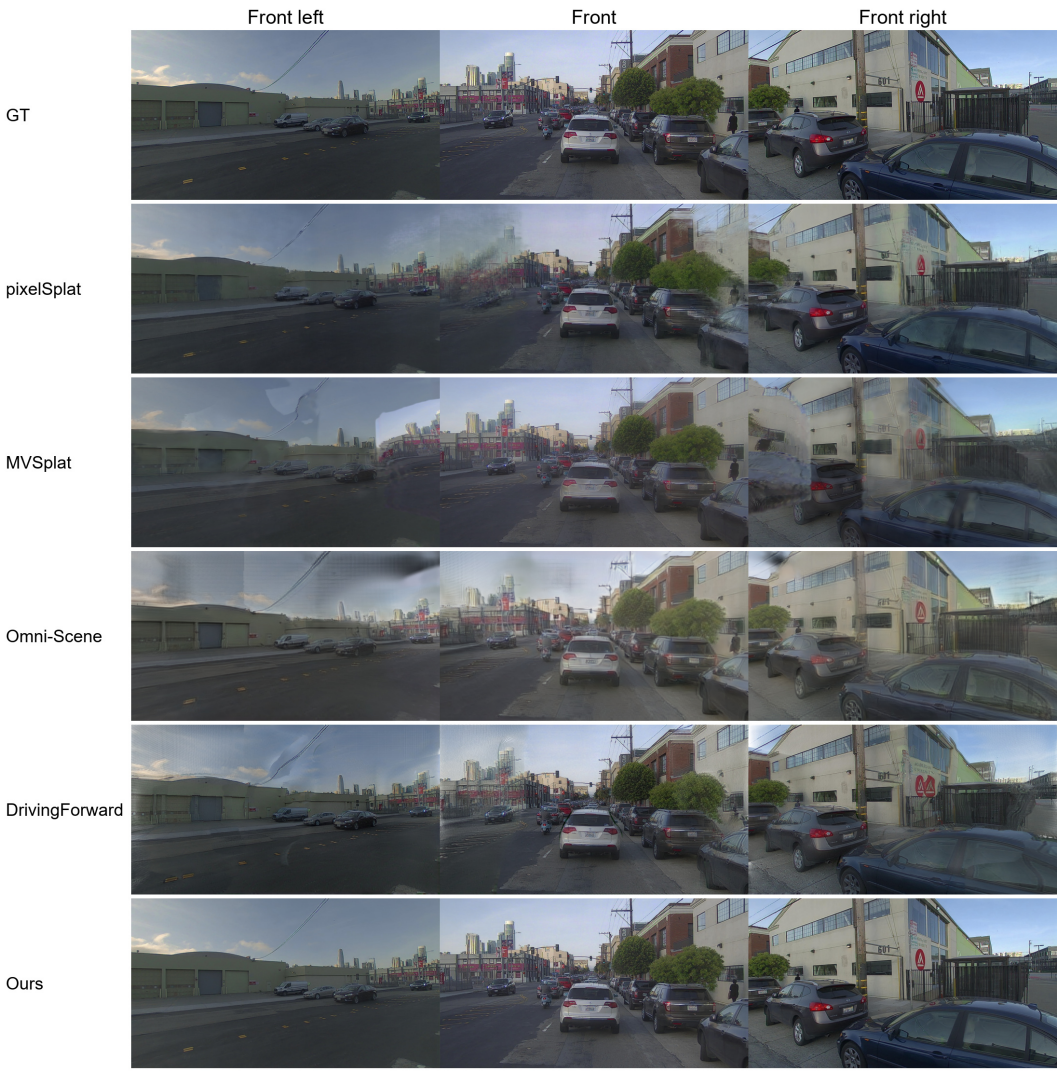

Figure 10: PandaSet zero-shot comparison, sequence 002. The second predefined visualization sample, using the same camera order, pretrained models, and evaluation protocol as Fig. 9. The samples were fixed before inference, not selected by reconstruction quality.

G Qualitative Transfer to a New Camera Rig

We additionally apply the nuScenes-trained model to a self-collected six-camera driving sequence without fine-tuning. This changes the camera configuration and scene distribution. Both methods use the same 32 synchronized six-camera observations and calibration, and render the same elevated virtual viewpoint. Figures 11 and 12 show the first, middle, and last frames of this sequence, rather than frames selected by a quality score. Each timestamp is reconstructed independently; this demonstration does not accumulate a scene over time.

Both methods are rerun with 352×640 inputs before network inference, and both visualizations are rendered at 560×1600. No ground-truth image is available at the virtual viewpoint, so this remains a qualitative transfer comparison. Road markings and vegetation boundaries are more distinct in parts of our renderings, while stretched surfaces and missing content remain visible in both methods. These examples illustrate transfer beyond the training camera rig without establishing a quantitative cross-dataset accuracy claim.

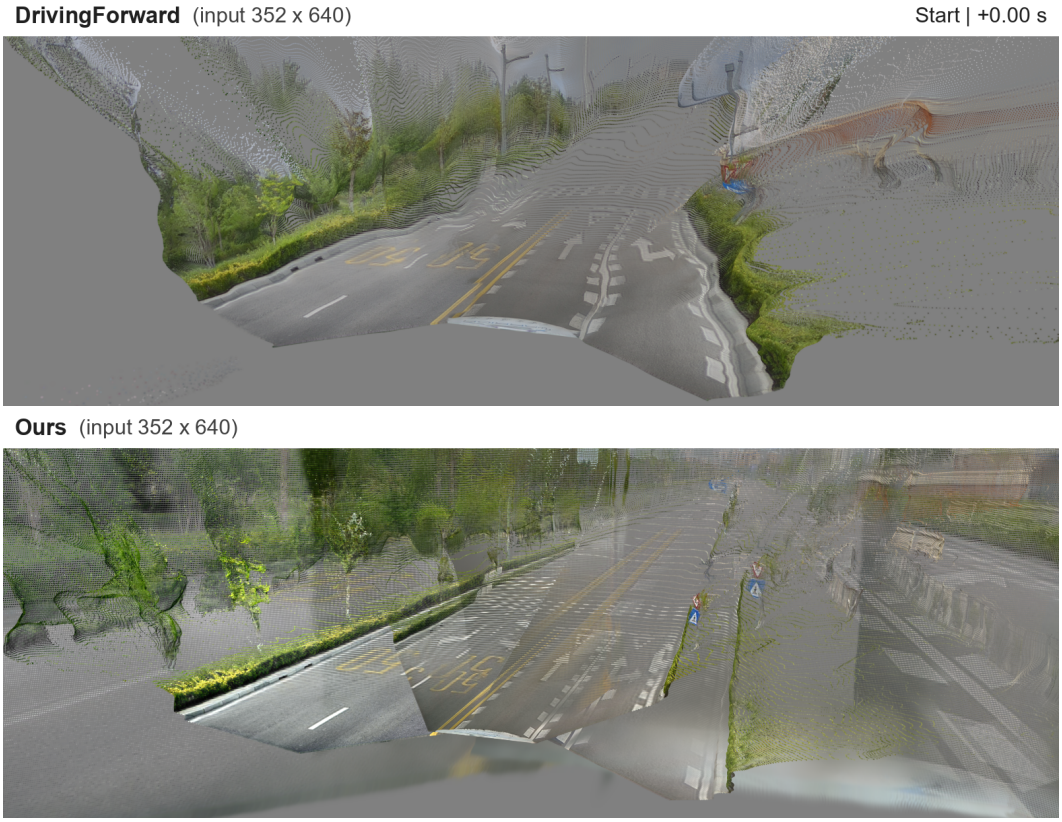

Figure 11: Qualitative transfer without fine-tuning. The first synchronized observation, viewed from an elevated virtual camera. DrivingForward is above Ours. Both methods retain their nuScenes-trained weights and use the same 352×640 input resolution.

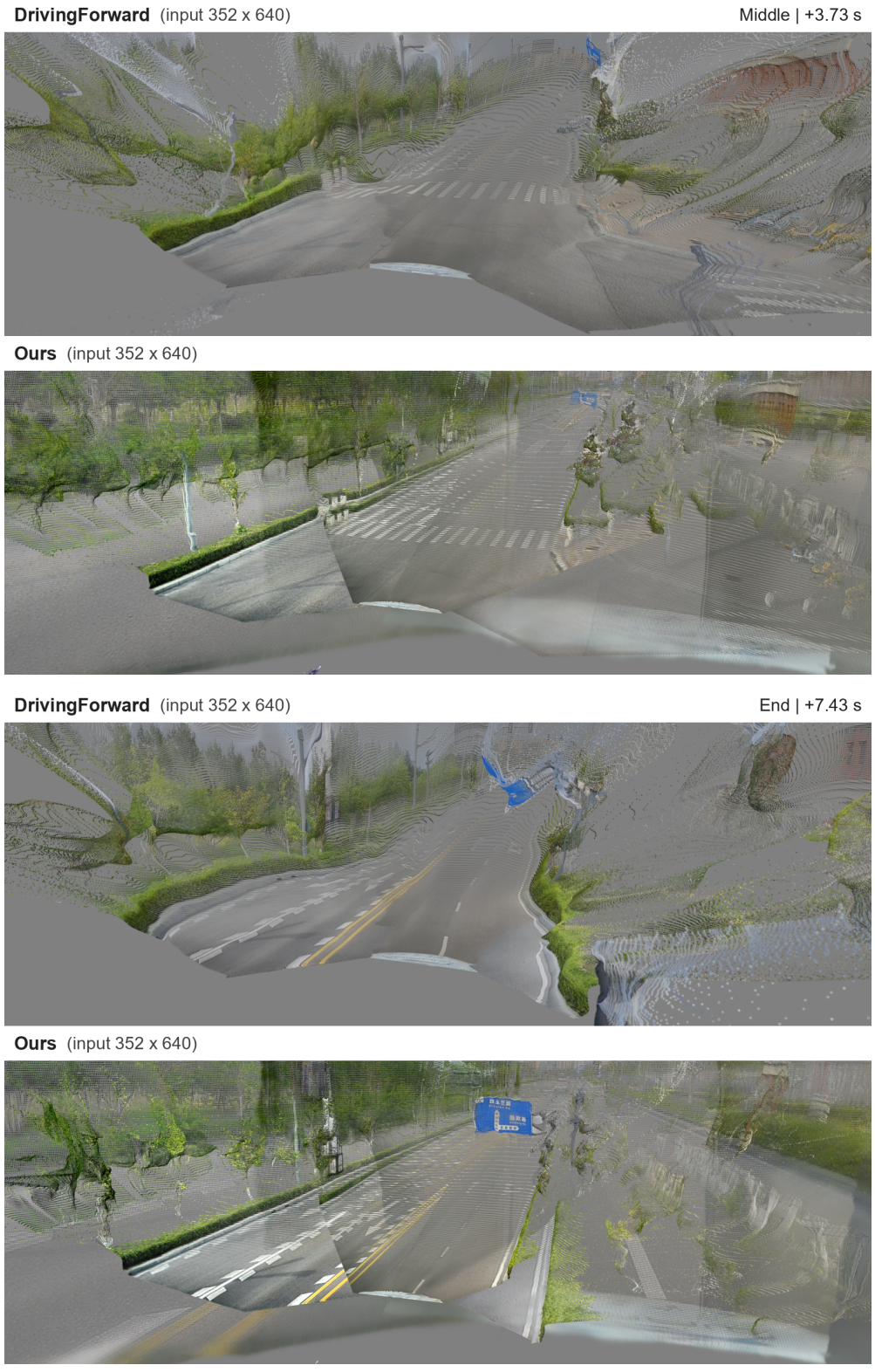

Figure 12: Further moments from the same unseen camera rig. Middle and final observations, approximately 3.73 and 7.43 seconds after Fig. 11. The camera viewpoint and method order are unchanged. Timestamps denote acquisition time, not video playback time.