

Supplemental Material

A Implementation Details

Initialization: Our pipeline begins by generating initial depth estimates using DepthSplat [21] (weight name: "depthsplat-gs-base-dl3dv-256x448-randview2-6-d94d996f.pth"), which processes input images and camera parameters to produce per-view depth maps. These depth maps are then unprojected into 3D space using camera intrinsics and extrinsics to form a dense point cloud. We initialize static Gaussians from this point cloud, and then perform low-resolution training at a resolution of 240×160 for three 3,000-iteration pruning epochs. Through our progressive pruning strategy, we gradually filter out less significant Gaussians based on opacity values (the threshold is computed from percentiles of all Gaussians), ultimately retaining a subset of 2×10^6 Gaussians. To mitigate the impact of initial depth inaccuracies, we defer the activation of the Depth Enhancer module until the 1,000th iteration after the first epoch. The final rendered depth maps from this initialization phase (denoted as \mathbf{D}^{prev}) serve as the foundation for subsequent full training processes. For Road Node initialization, we use the road mask to identify the corresponding Gaussians and assign them to the Road Node.

Training Details: The model contains 60,000 iterations of training, divided into 40,000 iterations for RGB-only training and 20,000 iterations for iterative Gaussian and depth enhancement training. The reason why we do not perform depth enhancement in the first 40,000 iteration is that, the early training stages typically produce meaningless depth and image outputs. There are severe floaters and artifacts resulting from the aggressive densification process. We follow the default learning rate settings for Gaussian properties as established in OmniRe [1]. Our proposed Road Node’s learning rate is 0.05. The Depth Enhancer module is activated halfway through training, with the diffusion process consisting of 80 steps and an early stop mechanism that prevents the smoothness loss from increasing for 8 continuous steps. For the warping loss, we utilize the views from 6 closest timestamp of the same camera ID. For quantitative comparisons, results for 3DGS, EmerNeRF, and S3GS are adopted from the S3GS [7] paper. All other methods were evaluated by us using their official codebases on a single NVIDIA H20 GPU. Key hyperparameters for our method are detailed in Tab. 4.

Table 4: Value of the key hyperparameters in our method.

Hyperparameter	Value	Description
$p \times p$	$\frac{H}{8} \times \frac{H}{8}$	Patch size for depth downsampling
λ_c	2.0	Confidence threshold in Depth Enhancer
λ_{ref}	1.0	Weight for reference depth loss in Depth Enhancer
λ_{smooth}	$\frac{1}{8}$	Weight for depth smoothness loss in Depth Enhancer
λ_w	$\frac{1}{16}$	Weight for warping loss in Depth Enhancer
N	20	Depth Enhancer update frequency
λ_{normal}	10.0	Weight for road normal alignment loss in Road Node
λ_{flat}	1.0	Weight for road flatness loss in Road Node
λ_{road}	0.1	Weight for Road Node loss

B Datasets and Baselines:

Our main experiments are conducted on the Waymo NOTR Dynamic32 [27] dataset, comparing our method against S3GS [7], PVG [8], OmniRe [1], and LiDAR-free baselines. The specific segment IDs are listed in Tab. 5. Our ablation studies are performed on a 12-sequence subset of the Waymo NOTR dataset, with segment IDs provided in Tab. 6.

Table 5: Segment IDs of Waymo NOTR Dynamic32 Dataset.

seg102319...	seg103913...	seg104444...	seg104980...	seg105887...	seg106250...	seg106648...	seg109636...
seg110170...	seg117188...	seg118463...	seg119178...	seg119284...	seg120278...	seg121618...	seg122514...
seg123392...	seg148106...	seg168016...	seg181118...	seg191876...	seg225932...	seg254789...	seg441423...
seg508351...	seg522233...	seg583504...	seg624282...	seg767010...	seg882250...	seg990779...	seg990914...

Table 6: Segment IDs of a 12 sequences subset used in our ablation experiments.

seg102319...	seg104980...	seg105887...	seg109636...	seg110170...	seg117188...
seg119284...	seg122514...	seg123392...	seg168016...	seg191876...	seg441423...

C Additional Results

In this section, we present more qualitative comparisons of our methods with others. Same as the main submission, we show the results of the LiDAR supervised methods (S3GS [7], PVG [8], OmniRe [1], and LiDAR-free methods (OmniRe w/o LiDAR, OmniRe + DepthSplat, ours).

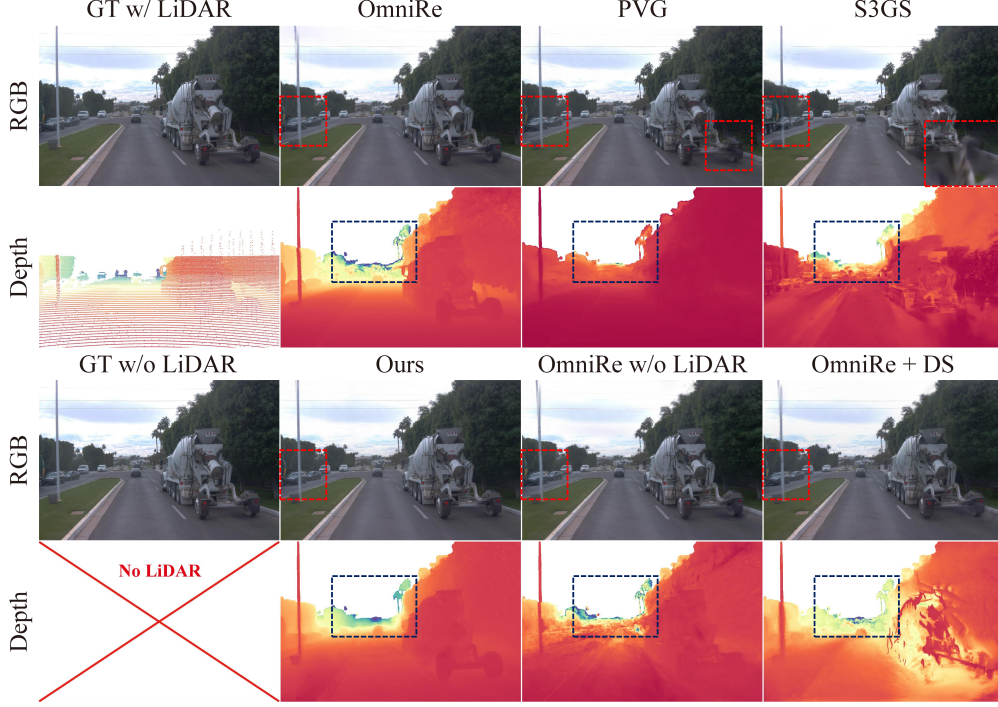


Figure 5: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

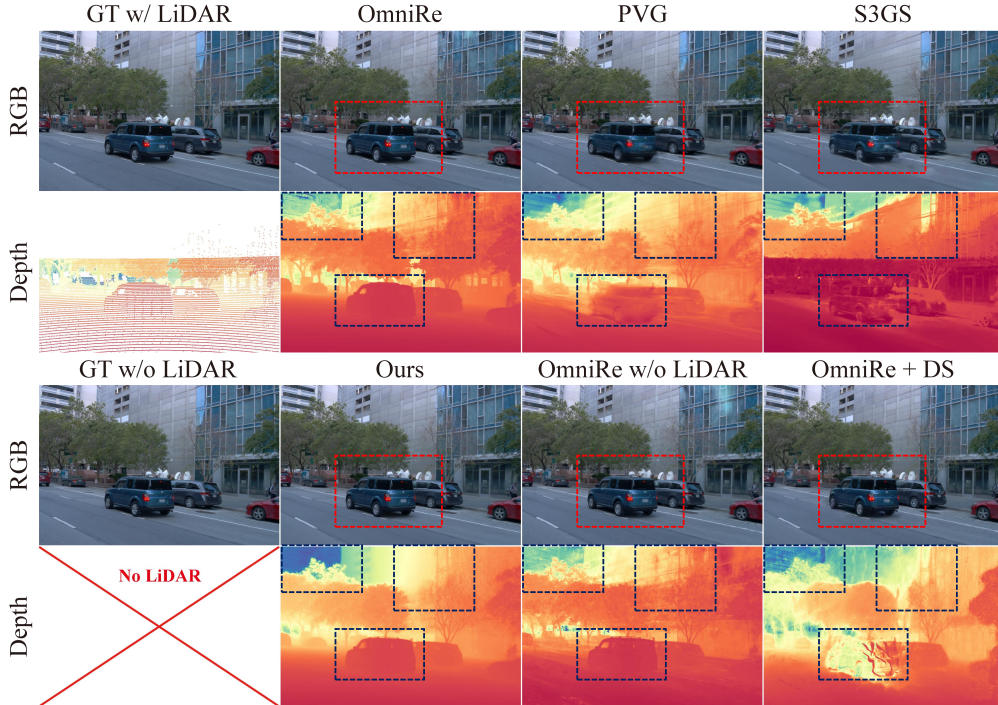


Figure 6: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

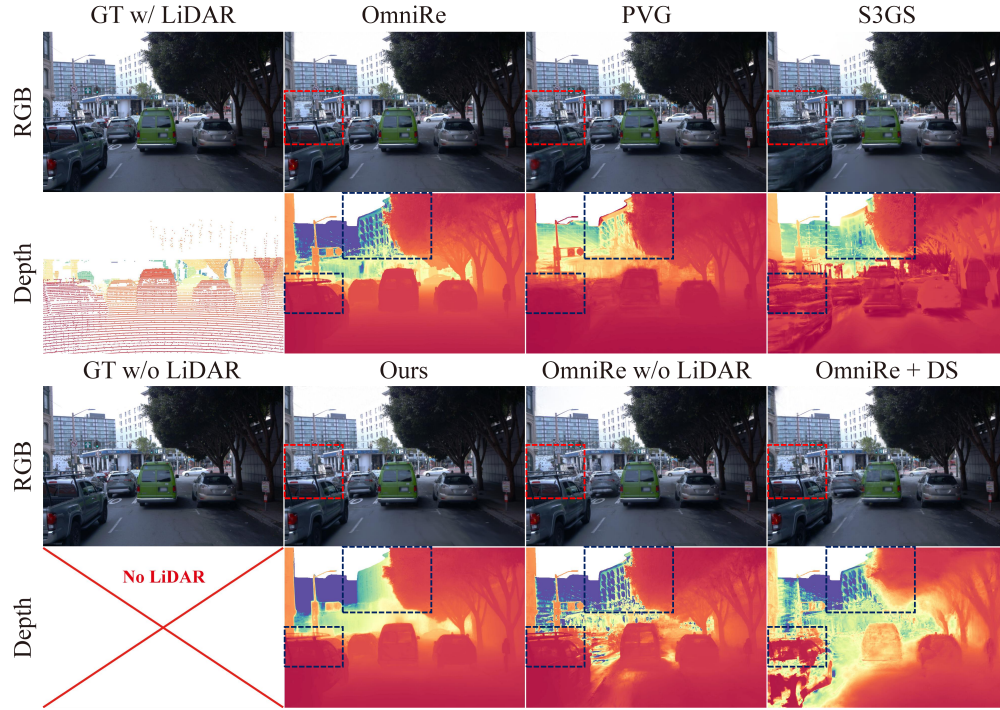


Figure 7: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

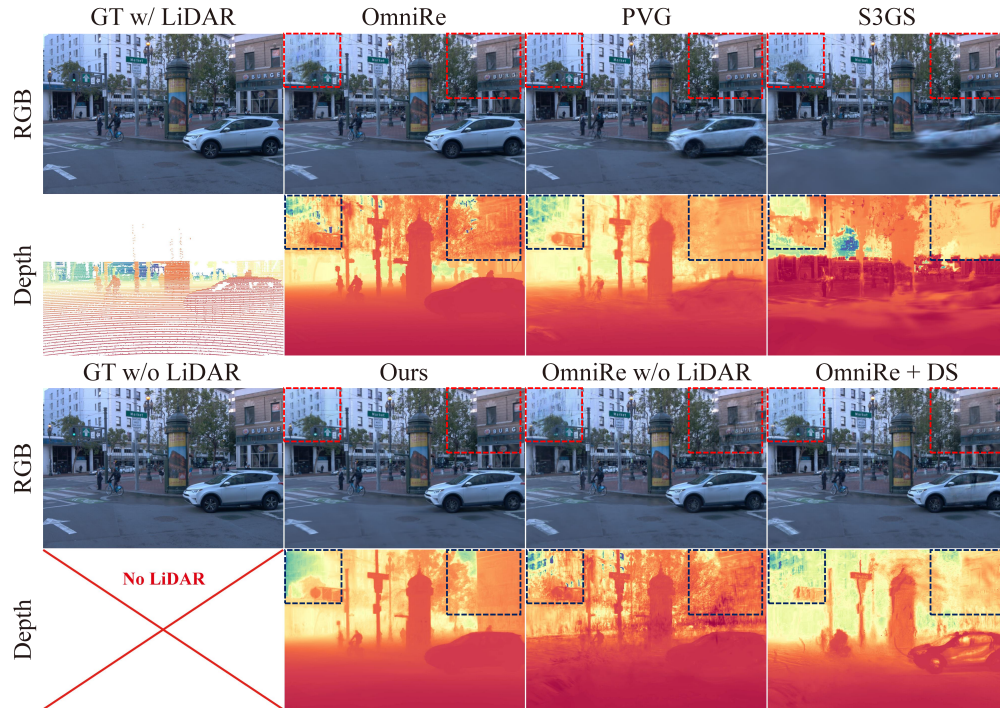


Figure 8: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

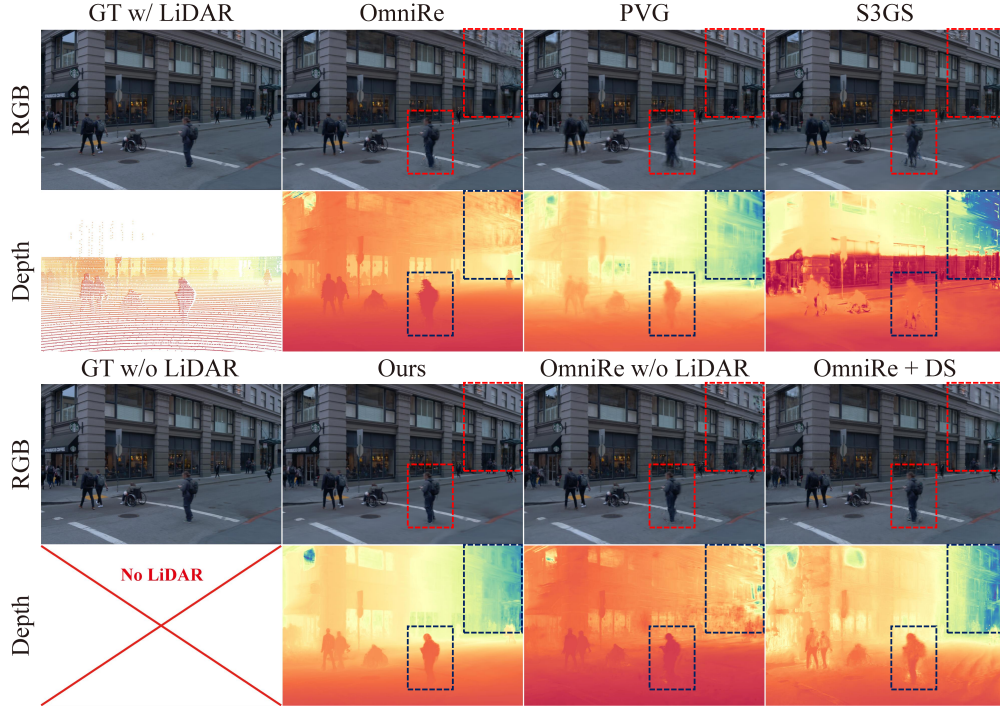


Figure 9: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

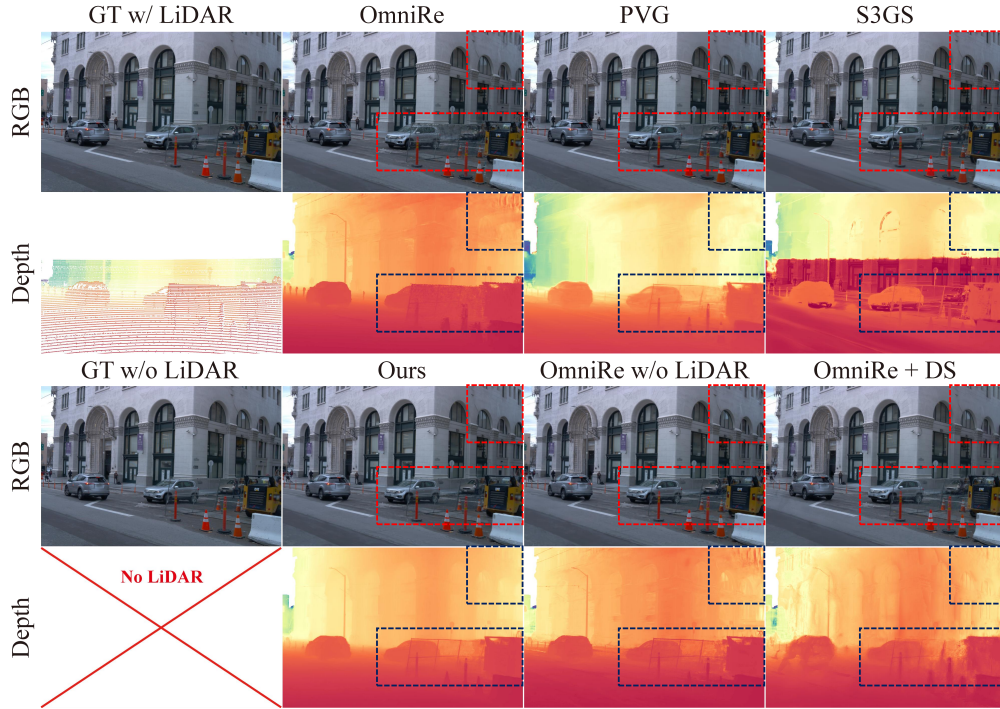


Figure 10: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

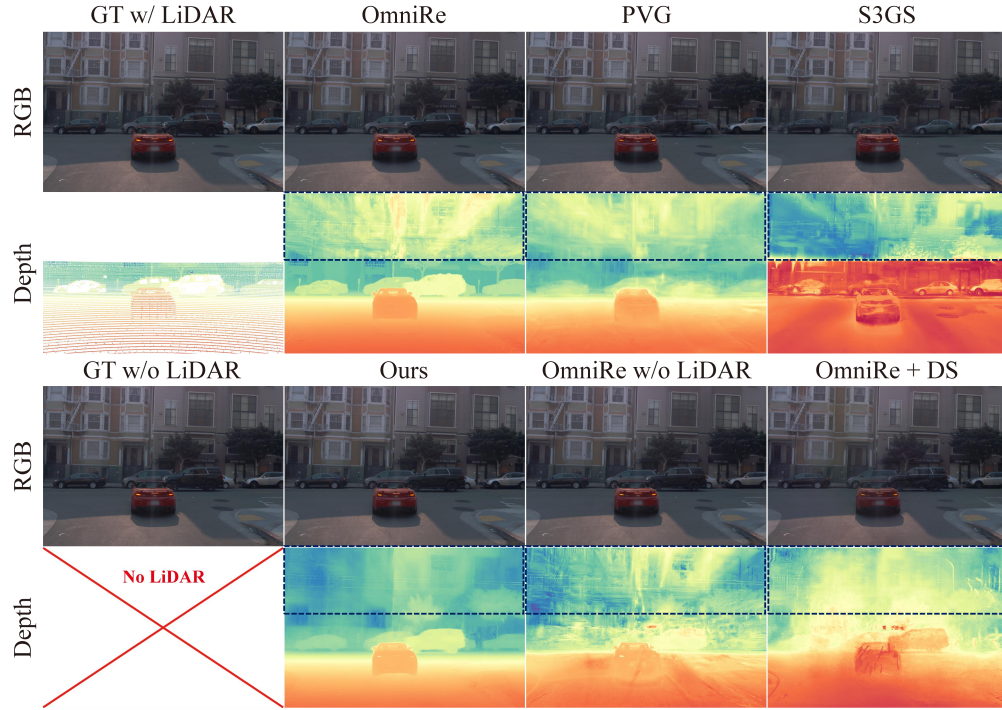


Figure 11: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

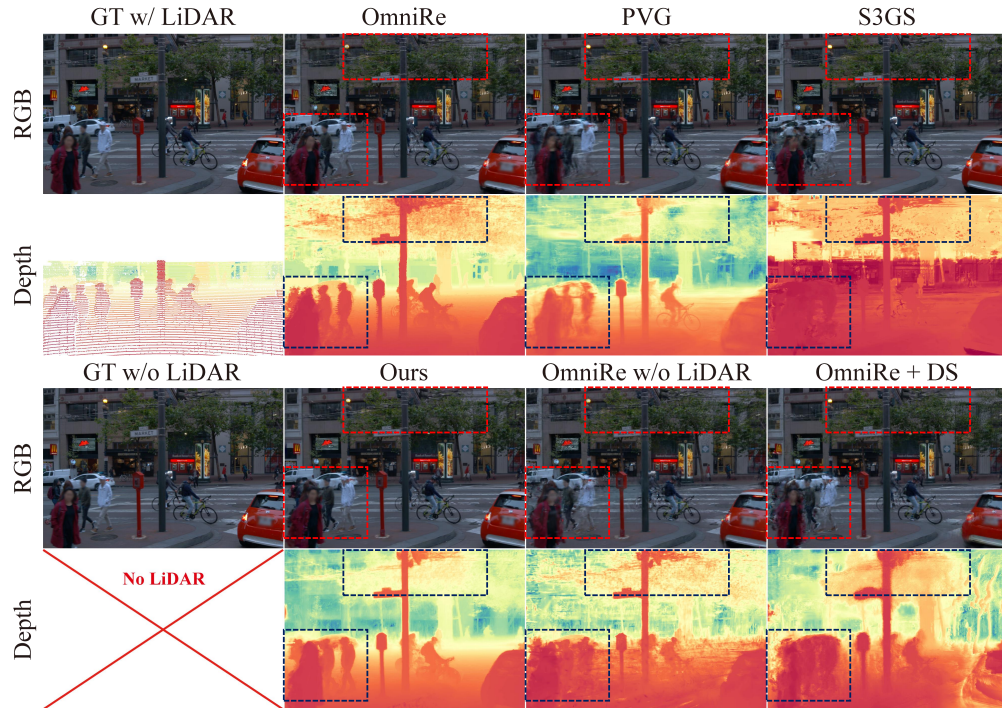


Figure 12: Comparison of our method with S3GS, PVG, OmniRe, and LiDAR-free baselines.

D Additional Experiments and Analyses

Summary. This section expands the computational analysis, clarifies assumptions about camera poses and our handling of dynamic objects, details sky modeling, strengthens ablations (including image-level metrics and road-node losses), visualizes depth evolution during training, and analyzes the initialization choices (SfM vs. MVS vs. monocular metric depth).

Notation. We use the main-paper notation throughout. “P.P.” denotes *Progressive Pruning*, “D.E.” denotes the *Depth Enhancer*, and “R.N.” denotes the *Road Node*. NVS denotes novel-view synthesis.

D.1 Computational Analysis of Progressive Pruning

Dense depth initialization yields an extremely large set of back-projected points. Training Gaussians on these points is both redundant and error-prone, because abundant low-accuracy points often produce artifacts. P.P. therefore serves two goals: (i) reduce computation and memory by early removal of inconsistent Gaussians and (ii) deliver a higher-quality initial point cloud that improves reconstruction.

Computational Analysis. Tab. 7 profiles memory, runtime, and the evolution from initial to retained training points over the first 30k iterations. The results show that when the initial input point cloud contains as many as 15M points, the Gaussian model retains around 6M points that are not effectively pruned. In contrast, our P.P. method prevents a significant waste of computational resources and avoids efficiency bottlenecks. Even with very large initial sets, P.P. can avoid out-of-memory errors and shorten training by eliminating redundant, inconsistent Gaussians before full optimization. In contrast, the original pruning method prunes primarily during rendering optimization, which defers these savings.

Table 7: Analysis of computational cost, comparing our P.P. method with a baseline using a manageable number of points to demonstrate the efficiency gains.

Method	Initial Points	Training Points [†]	GPU RAM	Runtime
w/o P.P.	15M	6M - 8M	34 GB	10 h
P.P.	2M	2M - 3M	20 GB	8 h

[†]Retained points used in optimization within the first 30k iterations.

D.2 The Dependency on Camera Pose Accuracy

Despite the LiDAR calibration issue which we aim to solve in our method, we assume accurate camera poses, consistent with common urban driving scene reconstruction settings and dataset priors.

To further address inaccurate camera poses, a common issue in data calibration, we propose the following possible solutions for future work. Specifically, when precise camera poses are unavailable, pose estimation algorithms, such as learning-based methods or traditional SfM, can provide initial estimates. Subsequently, a joint optimization module for camera poses can be incorporated during reconstruction, enabling simultaneous refinement of both the poses and the reconstructed scene.

D.3 Modeling Dynamic Objects

Our method can handle dynamic objects properly through the following aspects:

Scene graph decomposition. We model and decompose dynamic objects using a Scene Graph, an approach that follows our baseline, OmniRe [1]. Each dynamic object is represented as an independent node within this graph, equipped with its own set of Gaussian models (for dynamic or rigid representations).

Depth refinement for dynamics. Initial MVS depths inevitably contain errors on moving objects. First, since our method uses given ground-truth camera poses from Waymo, dynamic objects do not affect pose estimation. Depth errors are therefore localized to these dynamic areas, with minimal

impact on the static background. Furthermore, during the iterative optimization of the Gaussians and the D.E., the depth of dynamic objects is further refined (as shown in Fig. 13). Specifically, this refinement is driven by both the photometric loss from Gaussian splatting and the accurate geometric supervision provided by our diffusion-based D.E.. This joint optimization loop is effective for both static and dynamic scene components, enabling correction of initial errors and more accurate depth prediction.

D.4 Detailed Ablations with Image-Level Metrics

We complement geometric ablations with image reconstruction and NVS metrics, with results shown in Tab. 8. The full model performs best across settings, corroborating Tab. 3 in the main paper: degraded depth quality typically yields poorer NVS. Notably, the Periodic Update variant attains high reconstruction quality but markedly worse NVS, suggesting overfitting to training-view photometry with stale depth. This also underscores the importance of our real-time iterative refinement.

Table 8: Image-level metrics for ablations (image reconstruction and NVS). “Update” denotes D.E. update scheme.

Components			Settings of D.E.				Image reconstruction			Novel view synthesis		
R.N.	P.P.	D.E.	L_{ref}	L_w	L_{smooth}	Update	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
×	✓	✓	✓	✓	✓	Real-time	32.67	0.939	0.087	27.67	0.854	0.139
✓	×	✓	✓	✓	✓	Real-time	32.25	0.932	0.099	27.17	0.847	0.149
✓	✓	×	×	×	×	Real-time	32.77	0.939	0.087	27.52	0.851	0.143
✓	✓	✓	✓	×	×	Real-time	32.57	0.940	0.086	27.29	0.852	0.140
✓	✓	✓	✓	✓	✓	Periodic	33.95	0.951	0.075	26.90	0.823	0.128
✓	✓	✓	✓	✓	✓	Real-time	33.63	0.947	0.079	28.04	0.863	0.130

D.5 Road Node: Loss Analysis

We ablate the two R.N. loss terms independently, and conduct experiments on a subset of Waymo dataset (sequence id: ‘031’, ‘035’, ‘084’, ‘089’, ‘102’, ‘111’, ‘222’, ‘382’). As shown in Tab. 9, both the loss L_{plane} and the loss L_{shape} contribute positively to a more accurate geometry (evaluated with depth results). The best performance is achieved when both are used together.

Table 9: Ablation of road-node losses on Waymo subset.

Method	L1 \downarrow	Abs. Rel. \downarrow	RMSE \downarrow	$\delta < 1.25 \uparrow$
w/o L_{plane}	2.7879	0.1395	5.0075	0.9270
w/o L_{shape}	2.6502	0.1211	5.0192	0.9320
Full loss	2.4245	0.1161	4.5535	0.9459

D.6 Depth Evolution Across Training

Fig. 13 visualizes the online refinement process by showing: (i) the Gaussian-rendered depth (D.E. input), (ii) the refined depth (D.E. output), and (iii) visualizations of this process at three checkpoints (initial, mid, and final training stages). We observe a clear removal of floating fragments, stabilization of large planar regions, and improved cross-view geometric consistency as the refinement progresses.

D.7 Initialization: SfM vs. MVS vs. Monocular

In this section, we give a more comprehensive analysis of different initialization methods. Existing depth estimation methods are inevitably imperfect and unsuitable to use as strict ground-truth for high-fidelity reconstruction. As discussed in the main paper, monocular methods often yield relative depth with unknown scale, while MVS methods can struggle in dynamic scenes. Our core contribution is a framework that *starts* from an imperfect *metric* depth prior and jointly optimizes geometry and depth. Metric initialization is crucial for scale-correct point-cloud seeding and for strong, multi-view-consistent supervision.

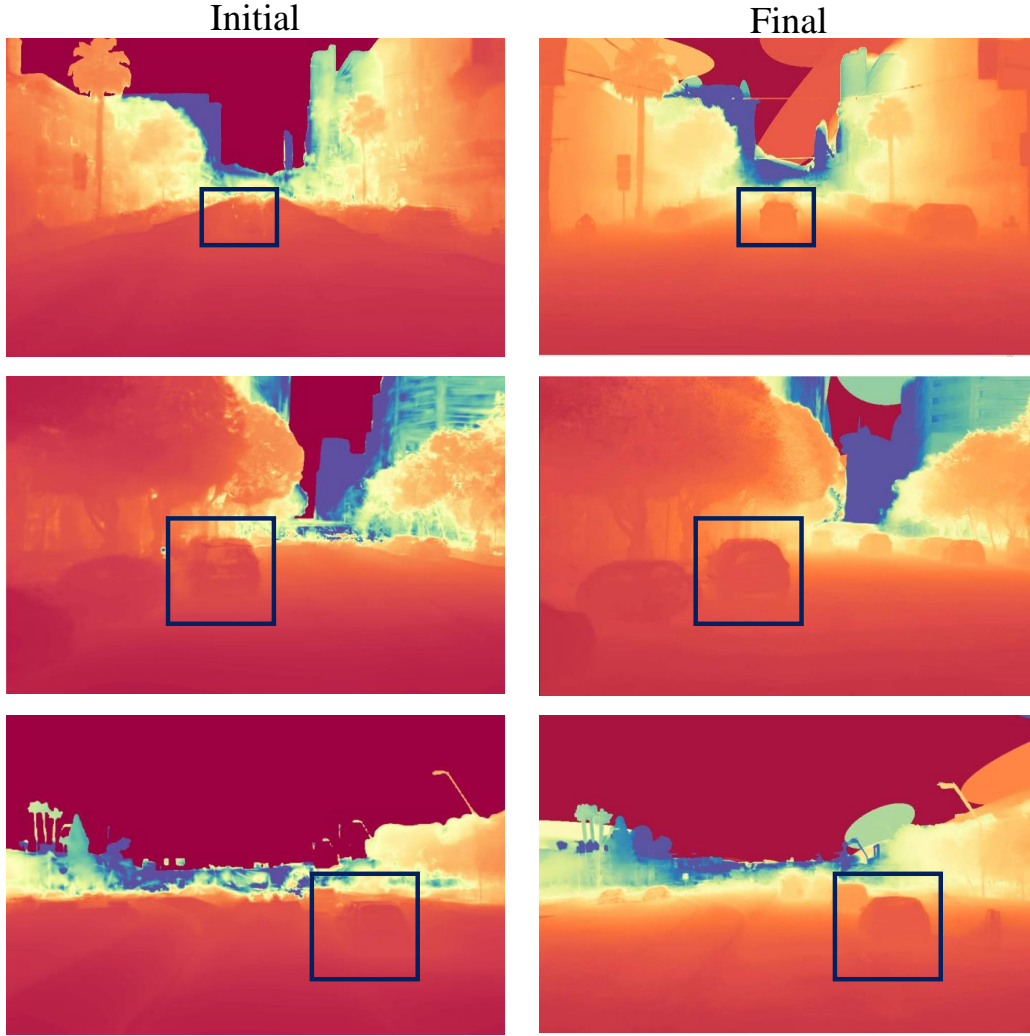


Figure 13: Depth evolution with online updates in three different stages: initial \rightarrow final.

Recent monocular depth estimation foundation models (e.g., UniDepthV2 [38] and Metric3D [39]) also report strong metric depth prediction performance. We adopt a DepthSplat-style MVS depth estimator in our main paper. Nevertheless, our pipeline is orthogonal to the depth source: any reasonable *metric* initializer can be used.

Monocular metric depth (UniDepthV2) in place of MVS. We replace the DepthSplat MVS method with UniDepthV2 [38] to initialize our pipeline. We conduct experiments on a subset of Waymo dataset (sequence id: '031', '035', '084', '089', '102', '111', '222', '382'), and Tab. 10 shows that our method still improves over baselines, confirming orthogonality to the initializer. Gains are smaller than with DS, likely because UniDepthV2 provides stronger initial depths and thus less headroom for refinement.

Why not SfM: a sparse point cloud initialization. We define dense initialization as back-projecting *pixelwise* metric depth maps into a redundant but coverage-complete point cloud. Sparse initialization denotes sampled point clouds (e.g., LiDAR, COLMAP), which provide only sparse supervision and may miss fine-scale or specific parts of the geometry.

In urban driving scenes, SfM still tends to be sparse due to limited overlap, textureless regions, and occlusions. Our contributions align naturally with dense initialization: P.P. compresses dense

Table 10: Monocular metric depth estimation as an initialization alternative on Waymo subset.

Methods	Image reconstruction			Novel view synthesis		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Omnire	33.96	0.955	0.057	28.50	0.874	0.117
Omnire+DS	31.30	0.936	0.078	28.11	0.876	0.110
Omnire+UniDepthV2	32.73	0.944	0.068	29.67	0.885	0.098
Ours+DS	34.35	0.954	0.068	29.75	0.883	0.107
Ours+UnidepthV2	34.39	0.954	0.069	29.92	0.891	0.096

initialization priors before full optimization, and D.E. requires dense targets for per-pixel supervision. However, with minor changes (e.g., drop P.P. and seed D.E. with reprojected sparse depths), the pipeline still applies to SfM sparse initialization.