LiHi-GS: LiDAR-Supervised Gaussian Splatting for Highway Driving Scene Reconstruction

Pou-Chun Kung[1,2*]   Xianling Zhang[1]   Katherine A. Skinner[2]   Nikita Jaipuria[1]

[1] Latitude AI   [2] University of Michigan, Ann Arbor

Figure 1. LiHi-GS provides higher quality color and depth renderings for interpolated novel views and for ego/actor shifts compared to state-of-the-art NeRF and GS-based methods [29, 35] LiHi-GS does particularly well on actor shifts at longer-ranges (183 meters) .

Abstract

Photorealistic 3D scene reconstruction plays an important role in autonomous driving, enabling the generation of novel data from existing datasets to simulate safety-critical scenarios and expand training data without additional acquisition costs. Gaussian Splatting (GS) facilitates real-time, photorealistic rendering with an explicit 3D Gaussian representation of the scene, providing faster processing and more intuitive scene editing than the implicit Neural Radiance Fields (NeRFs). While extensive GS research has yielded promising advancements in autonomous driving applications, they overlook two critical aspects: First, existing methods mainly focus on low-speed and feature-rich urban scenes and ignore the fact that highway scenarios play a significant role in autonomous driving. Second, while LiDARs are commonplace in autonomous driving platforms, existing methods learn primarily from images and use Li-DAR only for initial estimates or without precise sensor modeling, thus missing out on leveraging the rich depth in-formation LiDAR offers and limiting the ability to synthesize LiDAR data. In this paper, we propose a novel GS method for dynamic scene synthesis and editing with improved scene reconstruction through LiDAR supervision and support for LiDAR rendering. Unlike prior works that are tested mostly on urban datasets, to the best of our knowledge, we are the first to focus on the more challenging and highly relevant highway scenes for autonomous driving, with sparse sensor views and monotone backgrounds. Visit our project page at: https://umautobots.github.io/lihi_gs

1. Introduction

While there has been a lot of recent progress in semi-supervised and weakly supervised deep learning, in practice, most vision tasks for automated driving still rely on supervised learning and often fail to generalize to unseen scenarios. No matter how big the size of the dataset, capturing long tails is impractical, and neglecting them can have devastating consequences in safety-critical applications [24]. Dataset diversity is thus key to successful real-world de-

---

*The project is done during his internship at Latitude AI

1

ployment. However, data collection and human labeling remain time-intensive and costly.

Synthetic data offers a cost-effective approach to enhance diversity and capture long tails [40]. One such source is a gaming-engine-based simulation (e.g., CARLA [8]), which provides perfect annotation for free but lacks realism. More recently, Neural Radiance Fields (NeRFs) [21] have emerged as a popular choice for photorealistic novel view synthesis and scene editing. Prior works have shown promising results in autonomous driving scenarios [13, 18, 23, 28, 29, 37]. However, due to the sampling step in NeRF, these approaches face limitations such as computationally intensive rendering and degraded rendering quality at longer distances [30], which are important for highway driving.

In contrast, 3D Gaussian Splatting's (GS) [16] explicit scene representation enables faster and improved rendering for larger scenes and longer distances, coupled with intuitive scene editing. Recent works have shown promising results for autonomous driving scenarios [17, 35, 41, 43, 44]. However, existing works are limited to urban environments even though autonomous driving applications extend far beyond city streets. In particular, highway scenarios form a major portion of the operating domain for commercial Level 2 and Level 3 Advanced Driver Assistance Systems (ADAS) and pose distinct challenges for scene reconstruction, such as sparse sensor viewpoints, uniform backgrounds, and repetitive patterns. All of these combined make geometry learning for highway scene reconstruction quite difficult compared to the feature-rich and lower-speed urban settings.

LiDAR offers dense and precise depth measurements that enhance 3D scene understanding in challenging highway scenarios where camera images suffer from sparse viewpoints and lack of features. However, existing works only superficially use LiDAR data either for initialization [17, 43] or basic positional alignment [10, 44] and, therefore, fail to generalize to LiDAR-sparse regions. More recent approaches attempt to leverage LiDAR depth measurements by projecting point clouds onto camera image planes for depth supervision [6, 12, 35]. However, they only make use of the LiDAR measurements that overlap with camera views, make a strong simplifying assumption that the LiDAR sensor is physically close to the cameras (see Figure 2), and fail to support LiDAR novel view synthesis. In summary, existing methods miss out on leveraging LiDAR data to its full potential.

To address these limitations, we propose LiHi-GS, the first GS method with explicit LiDAR sensor modeling. It not only enables LiDAR supervision during training, resulting in significantly improved scene geometry learning and novel view image rendering quality, but it also provides the ability for realistic LiDAR synthesis. While prior works

have primarily focused on urban close-range scene reconstruction and editing (0-50 meters) [29, 35], we bridge this research gap by conducting comprehensive evaluations on highway scenes with objects at 200 meters and beyond, where the benefits of LiDAR supervision become particularly evident (see Figure 1). Our key contributions are as follows:

- Developed the first differentiable LiDAR rendering model for GS, enabling both LiDAR rendering and supervision.
- Demonstrated the importance of LiDAR supervision in GS for image and LiDAR novel view synthesis.
- LiHi-GS outperformed state-of-the-art (SOTA) methods in both image and LiDAR synthesis, particularly for view interpolation, ego-view changes, and scene editing tasks.
- Presented the first comprehensive study on highway scene reconstruction and editing, addressing a crucial yet underdeveloped use case in existing research.

2. Related Works

2.1. NeRF-Based Driving Scene Synthesis

NeRF [21] was originally designed for static room-scaled scenes. Follow-up works like Block-NeRF [26] extended it to city-scale reconstruction but are also limited to modeling static backgrounds. To model dynamic actors, Neural Scene Graph (NSG) [23] and MARS [33] create a scene graph in which each actor is modeled as an independent NeRF model and annotated 3D poses are used for scene composition. SUDS [31] and EmerNeRF [36] are SOTA unsupervised dynamic scene modeling methods, i.e. they do not rely on annotated poses. However, they do not support LiDAR rendering, which is an indispensable sensor in many autonomous driving systems.

2.2. LiDAR-Integrated NeRFs

Many recent works extend the NeRF formulation for LiDAR supervision [2, 14, 25]. Others focus on LiDAR-only novel view synthesis, like LiDAR-NeRF [28] and NFL [13]. DyNFL [32] and LiDAR4D [42] further extend it to dynamic scenes. UniSim [37] follows NSG-style dynamic

Figure 2. (Left) Issues with LiDAR projected pseudo-depth image supervision, highlighted in blue boxes. Points from both near and far distances can map to the same pixel, resulting in depth ambiguity. In the rendered opacity view (right), vehicles appear distorted in the case of pseudo-depth supervision, whereas LiHi-GS (middle) preserves object geometry and integrity.

Figure 3. System overview. LiHi-GS takes multiple cameras, LiDAR, and annotated 3D poses as input. In the preprocessing step, a LiDAR map combined with a COLMAP sparse point cloud is used for static scene initialization, while aggregated LiDAR points are used for dynamic object initialization. Our method enables LiDAR supervision during training and supports rendering both images and LiDAR.

scene modeling and supports both images and LiDAR supervision. NeuRAD [29] further addresses multiple approximations in UniSim and models camera rolling shutter and LiDAR ray drop and intensity for more realistic camera and LiDAR view synthesis.

2.3. GS-Based Driving Scene Synthesis

3DGS [16] explicitly represents scenes with Gaussians. The CUDA-accelerated Gaussian projection and rasterization expedited training and also enabled real-time rendering. PVG [4] is one of the first GS methods to reconstruct autonomous driving scenes using periodic vibration-based temporal dynamics. Recent works [17, 35, 43, 44] extend the NSG idea to GS and show promising results for driving scenarios. Unsupervised learning has also been investigated [12].

2.4. LiDAR-Integrated GS

In contrast to NeRFs, LiDAR measurements can be integrated into GS in multiple ways, of which the most common is initializing Gaussians with the LiDAR point cloud as a geometric prior [17, 43]. However, this approach fails to fully integrate LiDAR data into the training pipeline, leading to potential inaccuracies in scene geometry as the model may overfit camera images used for training. Correctly incorporating LiDAR data into the training regime is challenging because the original GS is designed to project 3D Gaussians onto a 2D image plane and then rasterize the image using depth-sorted 2D Gaussians. This 3D-to-2D camera projection is inherently incompatible with the LiDAR sensor model. To tightly integrate LiDAR measurements into GS training, Some studies indirectly supervise the GS model with a LiDAR-supervised NeRF model [41] or a LiDAR point cloud map represented by Gaussian Mixture Model [15]. However, this indirect training fails to provide LiDAR rendering capability. Alternatively, some works introduce an additional loss to encourage Gaussian centers to align with the LiDAR point cloud [10, 44]. This approach,

however, assumes that the LiDAR point cloud fully covers the camera-visible region, resulting in degraded image quality in areas without LiDAR measurements. More recent methods attempt to integrate LiDAR data directly into the training pipeline by projecting the LiDAR depth point cloud onto the camera image plane to create a pseudo-depth image for supervision [6, 12, 35]. However, this approach has limitations. First, the pseudo-depth image assumes depth measurements originate from the camera view, while in autonomous driving setups, LiDAR sensors are typically offset from the cameras. The side effect is shown in Figure 2. Second, this method fails to leverage LiDAR points outside the camera's field of view. To correctly and fully leverage LiDAR supervision in GS training, we propose a differentiable LiDAR model for GS that projects 3D Gaussians onto LiDAR range image frames. There is a concurrent work [3] on LiDAR modeling focused exclusively on LiDAR data synthesis.

3. Method

Figure 3 illustrates an overview of the method. Section 3.1 presents the 3D Gaussian scene representation for dynamic scenes and introduces the proposed LiDAR visibility rate. Section 3.2 discusses the camera modeling convention. Finally, Section 3.3 describes the novel LiDAR modeling framework with four key components: (1) a differentiable LiDAR rendering method to project 3D Gaussians into a range image frame; (2) 2D Gaussian scale compensation to better approximate LiDAR ray-tracing with image projection; (3) depth uncertainty rendering to remove floating artifacts on object edges; and (4) a decoupled camera-LiDAR pose optimization for handling temporal offsets between camera and LiDAR measurements for high-speed actors.

3.1. Dynamic 3D Gaussian Scene Representation

GS models the scene as a set of 3D Gaussians. Splat model $\mathbf{G}$ with N Gaussians is composed of the mean μ, rotation

3

quaternion q, scaling vector S, opacity α, spherical harmonic (SH) coefficients sh, and LiDAR visibility rate γ:

$$\mathbf{G} = \{G_i : (\mu_i, q_i, S_i, \alpha_i, sh_i, \gamma_i) \, | \, i = 1, ..., N\}. \quad (1)$$

Each Gaussian's covariance is parameterized by $\Sigma = RSS^T R^T$, where $S \in \mathbb{R}^3$ is a 3D scale vector with square roots of Σ's eigenvalues and $R \in \text{SO}(3)$ is the rotation matrix computed from quaternion q.

LiDAR Visibility. In addition to opacity α denoting the opaque state in an image, a LiDAR visibility rate γ is introduced to handle the fundamental differences in how the two sensors perceive the environment. For example, LiDAR reflects off surfaces based on their material and geometry. Low-reflective or semi-transparent surfaces can be missing in LiDAR measurements. Moreover, LiDAR has a limited sensing range compared to cameras. For example, the VLS-128 LiDAR can perceive only 5% of targets ¿ 180m, whereas high-resolution cameras can capture many more distant objects. Therefore, camera and LiDAR visibility are decoupled for each Gaussian via $\alpha_i^{lidar} = \alpha_i \gamma_i$ to better handle the challenging open highway scenes with many targets at far distances.

Gaussian Scene Graph. Dynamic driving scenes are reconstructed by separating the background and actors' splat models following the idea first proposed in [23] and later used in [17, 35, 43, 44]. The time sequence of actor transformations is then used to combine the background and actor splat models to create the full scene splat model:

$$\mu_{ij} = R_j \mu_{ij}^o + T_j \quad (2)$$

$$\Sigma_{ij} = R_j \Sigma_{ij}^o R_j^T \quad (3)$$

where Σ_{ij}^o, μ_{ij}^o are the ith Gaussian mean and covariance in the jth object model, where R_j and T_j are object rotation and translation.

3.2. Camera Modeling for Gaussian Splatting

Color Rendering. To render an image from a camera view from the Gaussian primitive, we need to project 3D Gaussians into a 2D image plane as follows:

$$\mu' = KW\mu \quad (4)$$

$$\Sigma' = JW\Sigma W^T J^T \quad (5)$$

here W is the camera pose with respect to the world frame, K is the camera intrinsic matrix, and J is the Jacobian of K used to project the 3D covariance into the image plane. Next, the pixel color is computed as:

$$\hat{C}(p) = \sum_{i \in N} c_i \alpha_i' \prod_{j=1}^{i-1} (1 - \alpha_j') \quad (6)$$

where p is the pixel in an image, and c_i is the color of a Gaussian obtained by sh_i and view direction v. The opacity, α_i', of Gaussian at pixel x' is:

$$\alpha_i' = \alpha_i \exp\left(-\frac{1}{2}(x' - \mu_i')^T \Sigma_i^{-1}(x' - \mu_i')\right) \quad (7)$$

where x' and μ' are the rendered pixel and Gaussian center in projected 2D image space.

Depth Rendering. Since cameras do not provide direct depth measurements, recent studies project LiDAR point cloud into the image plane to generate pseudo-depth images for GS training [6, 12, 35]. The rendered depth for each pixel is computed as:

$$\hat{D}(p) = \sum_{i \in N} d_i \alpha_i' \prod_{j=1}^{i-1} (1 - \alpha_j') \quad (8)$$

where d_i is the depth of the Gaussian center.

3.3. LiDAR Modeling for Gaussian Splatting

3.3.1. Range Image Rendering

To render LiDAR range images from 3D Gaussians, we first convert 3D Gaussian means from the original Cartesian coordinates to spherical coordinates:

$$r = \sqrt{x^2 + y^2 + z^2}, \quad (9)$$

$$\theta = \arctan 2(y, x), \quad (10)$$

$$\phi = \arcsin\left(\frac{z}{r}\right) \quad (11)$$

Next, we project 3D Gaussians into the LiDAR range image frame. For a LiDAR with N beams and inclination $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$, a coordinate in the range image $\mathcal{R} \in \mathbb{R}^{N \times W}$ is represented as (v, u) where v is the index of the closest element in Φ and $u = \frac{\theta W}{2\pi}$. A LiDAR beam is approximated as a frustum of a range image pixel. This holds true only for beams with low angle resolution. While most LiDARs have a relatively sparse beam distribution on the edge of the vertical Field-of-View (FOV), we filter out Gaussians in which the center is too far off from the beam angle: $|\phi_{closest} - \phi| > 0.5°$.

The same idea is used to project the 3D covariance matrix Σ by first converting it to spherical space:

$$\Sigma_{\text{spherical}} = J\Sigma J^T \quad (12)$$

where the Jacobian is:

$$J = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial z} \\ \frac{\partial \phi}{\partial x} & \frac{\partial \phi}{\partial y} & \frac{\partial \phi}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{x}{r} & \frac{y}{r} & \frac{z}{r} \\ \frac{-y}{x^2+y^2} & \frac{x}{x^2+y^2} & 0 \\ \frac{-xz}{r^2\sqrt{r^2-z^2}} & \frac{-yz}{r^2\sqrt{r^2-z^2}} & \frac{\sqrt{r^2-z^2}}{r^2} \end{bmatrix} \quad (13)$$

We then project into the LiDAR range image space:

$$\Sigma_{uv} = A\Sigma_{\text{spherical}} A^T \quad (14)$$

4

Figure 4. LiDAR depth uncertainty rendering helps compensate for noisy artifacts around object edges.

Figure 5. Depth uncertainty filter removes floating artifacts from rendered point cloud.

Figure 6. Camera-LiDAR misalignment for highway actors.

Given the non-uniform LiDAR beam inclination, the Jacobian A depends on the elevation angle resolution:

$$A_i = \begin{bmatrix} \frac{1}{\Delta\theta_i} & 0 \\ 0 & \frac{W}{2\pi} \end{bmatrix} \quad (15)$$

where $\Delta\theta_v = \frac{1}{2}(\theta_{v+1} - \theta_{v-1})$. Finally, we can sort Gaussians by the distance of their center r and rasterize a rendered range image $\hat{\mathcal{R}}$ with the same depth rendering equation as (7) and (8).

3.3.2. 2D Gaussian Scale Compensation

In the original GS image rasterization model, a small Gaussian far from the camera image plane becomes invisible in the projected pixel due to numerical instability. However, since LiDAR emits laser beams to measure the distance of objects, a Gaussian should remain fully visible if its center is sufficiently close to the ray, regardless of its scale. To address Gaussians that are lost during projection, we rescale the 2D Gaussian when it is near the ray and has a scale below the visible threshold. Eigendecomposition is applied to Gaussians to get their 2D scales after projection:

$$\Sigma_{uv}^i = V_i S_{uv}^i V_i^{-1}, \quad S = \text{diag}(s_1, s_2) \quad (16)$$

Visible scale depends on the distance of the Gaussian:

$$s_\delta^i = 2d_i \tan\left(\frac{\pi}{w}\right) \quad (17)$$

We adjust the invisible scale that is too small:

$$\tilde{s}_{ij} = \begin{cases} \frac{s_\delta^i}{3}, & \text{if } s_{ij} < \frac{s_\delta^i}{3} \\ s_{ij}, & \text{otherwise} \end{cases} \quad (18)$$

Finally, the 2D covariance matrix is constructed using the updated scale.

$$\tilde{\Sigma}_{uv}^i = V_i \, \text{diag}(\tilde{s}_{i1}, \tilde{s}_{i2}) V_i^{-1} \quad (19)$$

3.3.3. Depth Uncertainty Rendering

We note that the point cloud rendered using GS can include floating artifacts on object edges. This is mainly because depth discontinuities are hard to represent using the inherently continuous Gaussian distribution. To render a crisp and clean point cloud with clear object edges, we additionally incorporate depth uncertainty to identify pixels that cause floating noise in the rendered image, as shown in Figure 4. A low-pass filter using an uncertainty threshold is then applied to filter out floating points. To integrate the uncertainty estimation into the CUDA rasterization, uncertainty is calculated as the incremental weighted variance along each ray.

$$W_n = W_{n-1} + \alpha_n' \quad (20)$$

$$\mu_n = \mu_{n-1} + \frac{\alpha_n'}{W_n}(d_n - \mu_{n-1}) \quad (21)$$

$$\Gamma_n = \Gamma_{n-1} + \alpha_n' \cdot \frac{W_{n-1}}{W_n}(d_n - \mu_{n-1})(d_n - \mu_{n-1})^T \quad (22)$$

where α_n' is the opacity contributed by a Gaussian, described in Eq. 7. Figure 5 demonstrates the rendered point cloud with and without the uncertainty filter.

3.3.4. Camera-LiDAR Actor Alignment with Decoupled Pose Optimization

Our method heavily relies on accurately labeled poses for actors' reconstruction. However, not only can human annotations be imperfect, the camera and LiDAR measurements can also be misaligned for fast-moving actors due to the temporal observation difference, as shown in Figure 6. Unlike [29, 35] which optimize a unified pose, we propose a

5

decoupled pose optimization to solve pose misalignment between sensors in a more agnostic way. Thus, the GS model for LiDAR and camera rendering is constructed separately via Eq. 2, 3 using different poses T_{lidar}^j and T_{camera}^j.

3.4. Training Losses

The total optimization objective is as follows:

$$\mathcal{L} = \mathcal{L}_c + \lambda_1 \mathcal{L}_{Lidar} + \lambda_2 \mathcal{L}_{opa}^L + \lambda_3 \mathcal{L}_{reg}^s + \lambda_4 \mathcal{L}_{opa}^c \quad (23)$$

$$\mathcal{L}_{opa}^c = \mathcal{L}_{sky} + \mathcal{L}_{fg} + \mathcal{L}_{reg}^{obj} \quad (24)$$

Color loss $\mathcal{L}_c$ follows the color image loss from [16]. LiDAR loss is calculated as the L1 difference between the target and rendered range images: $\mathcal{L}_{Lidar} = \|\mathcal{R}_{img} - \hat{\mathcal{R}}\|_1$. $\mathcal{L}_{opa}^L$ encourages LiDAR visibility $\alpha_{lidar} = 1$ for pixels with depth returns in the range image. This loss term is crucial considering LiDAR's 360° field of view, which includes regions outside camera coverage. $\mathcal{L}_{reg}^s$ is a scale regularization term designed to make Gaussians more evenly shaped, preventing spike artifacts and providing more accurate depth rendering, inspired by [34]. $\mathcal{L}_{opa}^c$ encompasses all opacity losses from the camera image. $\mathcal{L}_{sky}$ ensures sky pixels have low opacity, while $\mathcal{L}_{fg}$ guides foreground pixels to have high opacity. Additionally, $\mathcal{L}_{reg}^{obj}$ is a regularization term used to improve decomposition effects, following [35]. Further details on each loss term are provided in the supplementary.

4. Experiments

4.1. Dataset

Existing research often relies on open-source datasets for evaluation that are dominated by texture-rich urban scenarios with dense sensor coverage. This limits their applicability to feature-sparse and view-sparse highway scenarios — a key use case for autonomous driving. To address this gap, we conduct experiments on a self-collected dataset with challenging highway environments. Our data collection vehicle is equipped with a VLS-128 LiDAR and three cameras facing front, back-left, and back-right, synchronized at 10 Hz. We gather highway data across three

U.S. states — Michigan, Pennsylvania, and North Carolina. Each data slice includes human-labeled bounding box annotations, enabling the construction of Gaussian scene graphs. Four challenging segments, ranging from 16 to 25 seconds in length and featuring multiple moving objects, high ego speeds, small scale far field objects, and monotonous backgrounds, are used in our experiments. Frames sampled every second are used for evaluation, and the rest are used exclusively for training.

4.2. Baselines

LiHi-GS is compared with four baselines. 1) Instant-NGP [22], an image-only scalable NeRF-based method with a feature hash grid. 2) NeuRAD [29], a SOTA NeRF-based method for driving scenes that supports LiDAR supervision and rendering. 3) 3DGS [16]; we use a more efficient re-implementation, Nerfstudio splatfacto-big [27, 38], higher quality than default splatfacto. Here, LiDAR point clouds are used with COLMAP sparse points for Gaussian initialization. 4) StreetGS [35], a SOTA GS-based method designed for autonomous driving. Here, LiDAR is used in initialization and depth image loss.

Ablation studies are conducted with our LiDAR loss disabled and replaced with the depth image loss from [6, 17, 35] in the proposed LiHi-GS pipeline.

4.3. Novel View Image and LiDAR Synthesis

Benchmarking. Table 1 shows the quantitative comparison of our method LiHi-GS with SOTA baselines for novel view rendering quality. LiHi-GS outperforms Instant-NGP and 3DGS on image metrics PSNR and SSIM by a large margin since both methods are not designed for dynamic scene reconstruction. However, in terms of LPIPS, 3DGS typically does the best while our method is second best. This reflects GS methods can easily overfit to the static background resulting in high LPIPS scores even in the absence of accurate actor modeling. When compared with SOTA methods for dynamic scene reconstruction (NeuRAD and StreetGS), LiHi-GS has the overall best image rendering quality. The only exception is the Greenville scene with

| Method | LiDAR | Pittsburg, PA | | | | | Dearborn, MI | | | | | Greenville, SC | | | | | Pittsburg II, PA | | | | |
| | | Image | | | LiDAR | | Image | | | LiDAR | | Image | | | LiDAR | | Image | | | LiDAR | |
		PSNR↑	LPIPS↓	SSIM↑	Mean↓	Med↓	PSNR↑	LPIPS↓	SSIM↑	Mean↓	Med↓	PSNR↑	LPIPS↓	SSIM↑	Mean↓	Med↓	PSNR↑	LPIPS↓	SSIM↑	Mean↓	Med↓
Benchmarking																					
Instant-NGP [22]	X	25.56	0.396	0.833	-	-	26.80	0.339	0.862	-	-	24.41	0.367	0.868	-	-	27.00	0.39	0.86	-	-
NeuRAD [29]	O	29.14	0.304	0.891	4.19	0.60	30.40	0.281	0.913	2.21	0.34	30.74	0.245	0.917	1.64	0.18	31.24	0.307	0.923	1.16	0.20
3DGS [16]	O	26.87	0.230	0.900	-	-	27.79	0.238	0.912	-	-	26.48	0.278	0.913	-	-	30.43	0.238	0.927	-	-
StreetGS [35]	O	29.65	0.313	0.907	5.02	4.92	29.98	0.302	0.924	8.23	3.60	26.68	0.337	0.920	15.99	6.72	31.48	0.330	0.931	5.57	2.77
LiHi-GS (Ours)	O	30.04	0.268	0.913	2.655	0.60	30.76	0.265	0.928	1.68	0.49	30.07	0.276	0.932	1.15	0.32	31.89	0.287	0.935	1.14	0.32
LiDAR Loss Ablation																					
Only $\mathcal{L}_{RGB}$	X	29.68	0.284	0.911	10.21	3.20	28.69	0.308	0.918	7.39	2.63	28.88	0.289	0.930	10.68	2.85	31.12	0.290	0.934	3.38	2.71
+ $\mathcal{L}_{depth}$ [6, 12, 35]	O	29.53	0.285	0.911	4.63	4.01	29.42	0.276	0.926	5.87	2.04	28.66	0.291	0.929	9.65	3.25	31.21	0.288	0.934	4.78	2.13
+ Proposed $\mathcal{L}_{lidar}$	O	30.04	0.268	0.913	2.655	0.60	30.76	0.265	0.928	1.68	0.49	30.07	0.276	0.932	1.15	0.32	31.89	0.287	0.935	1.14	0.322

Table 1. Comparison of image and LiDAR novel view rendering. Best and second best results are highlighted in red and orange.

6

Figure 7. Comparison of the rendered depth image from LiHi-GS vs. SOTA baselines. Results show cleaner geometry over StreetGS due to the added LiDAR supervision. NeuRAD depth also has overall correct geometry but is noisier. Using a 500m depth threshold to classify the sky (white) showed that NeuRAD introduces noisy depth measurements on the sky and fails to model the distant mountain.

*Note that StreetGS is doing LiDAR depth image supervision, and the point cloud is rendered using our LiDAR rendering method. This indicates the scene geometry can be wrong with depth image supervision.

Figure 8. Point cloud synthesized using different methods. Our method (LiHi-GS) provides a clean point cloud with minimal noise on objects' edges.

Figure 9. Comparing LiDAR range images from a model trained with the proposed LiDAR loss vs. existing depth image loss.

the vehicle driving down a wide lane with most of the things within camera view lying outside the LiDAR sensing range. Since GS relies on good point cloud initialization, this scene is challenging for all GS-based methods vs. NeRF-based NeuRAD. However, this issue can potentially be solved by advanced GS with activate densification [1] or dense initialization [9]. Full qualitative image rendering results are included in the supplementary. Qualitative rendered depth is shown in Figure 7. LiHi-GS captures finer geometric details, accurately rendering the sky and distant mountain.

In terms of LiDAR rendering performance, StreetGS [35] has the worst quality since their depth image supervision does not leverage the full geometry information from LiDAR and is not designed for LiDAR

rendering. On the other hand, LiHi-GS consistently has the lowest L1 mean error due to the decoupled pose optimization and uncertainty filter that is lacking in NeuRAD, as shown in Figure 8. We also discover the same or slightly higher LiDAR L1 median error than NeuRAD. We assume the main cause is that NeuRAD's ray tracing depth rendering is more accurate than our current projection-based approach. While the differences in LiDAR L1 median metrics between NeuRAD and LiHi-GS are small, the actor/sensor shift experiments demonstrate that LiHi-GS maintains more robust rendering quality under novel viewpoints. This suggests that despite similar quantitative performance on reconstruction metrics, LiHi-GS exhibits better dynamic scene composition to novel conditions (Figure 10 and 11).

LiDAR Loss Ablations. We also compare the impact of depth image loss used in prior works and our proposed loss. The results indicate depth image loss does not improve the image rendering quality and can sometimes degrade the image quality, but it does learn more accurate geometry, which leads to a lower LiDAR L1 mean error. In contrast, our LiDAR supervision not only improves the image rendering quality but also provides accurate LiDAR rendering. The LiDAR range image is visualized in Figure 9.

4.4. Scene Editing With Ego and Actor Shifting

We further evaluate the rendering quality with ego-vehicle shifting and surrounding actors shifting. To obtain views different from the original trajectory, we apply lateral shifting for both ego and actors from 1 meter to 3 meters. Since the ground truth after scene editing does not exist, we follow [29] to report the FID score to analyze the data synthesis quality. Table 2 shows our LiHi-GS generates the most realistic image from new viewpoints that are off from the original trajectory and learns a better representation of actors. The qualitative results of actor shifting and ego shifting are shown in Figures 10 and 11.

7

Figure 10. Image renderings with lateral actor shift. LiHi-GS has the best overall image quality, given the accurate geometry information learned from LiDAR supervision.

Figure 11. Image rendering with lateral ego-vehicle shift. LiHi-GS reconstructs distant vehicles with correct color modeling, better shadow shape, and fewer ghost artifacts.

	Ego lateral shift			Actor lateral shift		
	1m	2m	3m	1m	2m	3m
NeuRAD	63.2	82.42	101.45	51.25	55.61	79.70
StreetGS	56.09	69.59	82.24	43.65	54.48	65.86
LiHi-GS	40.70	57.47	80.69	30.91	43.72	54.50

Table 2. FID (↓) scores when shifting ego vehicle or actors pose

$\mathcal{L}_{opa}^{L}$	$\mathcal{L}_{reg}^{s}$	Decoupled Pose Opt.	Scale Comp.	LiDAR Vis.	Image			LiDAR
					PSNR↑	IPIPS↓	SSIM↑	Mean↓
X	O	O	O	O	30.54	0.229	0.927	2.94
O	X	O	O	O	29.65	0.234	0.926	2.76
O	O	X	O	O	30.49	0.228	0.927	2.67
O	O	O	X	O	30.22*	0.245*	0.923*	2.71*
O	O	O	O	X	30.59	0.228	0.928	2.67
O	O	O	O	O	31.63	0.208	0.935	2.61

*Disabling 2D scale compensation caused an exponential increase in memory usage, leading the job to crash midway.

Table 3. Ablation Studies.

4.5. Ablations

Table 3 presents the quantitative results when removing the different proposed designs from our pipeline.

Removing LiDAR opacity loss $\mathcal{L}_{opa}^{L}$ leads to the highest LiDAR L1 mean error. The visual result is shown in supplementary. Scale regularization $\mathcal{L}_{reg}^{s}$ also has a great impact on both image and LiDAR quality. Render depth without scale regularization is inaccurate and can degrade the image quality during LiDAR supervision. Disabling decoupled pose optimization leads to camera-LiDAR misalignment and worse PSNR rendering quality. The model without 2D scale compensation crashed halfway through the training due to the memory overhead. This is because, without 2D scale compensation, the LiDAR supervision tends to increase the size of Gaussians at a far distance to fill in the projected LiDAR pixel, which contradicts the real geometry of the scene. In the end, LiDAR visibility significantly improves both image and LiDAR rendering quality. This reflects the fundamental differences between the two sensors (Section 3.1). Also, the rendered depth in GS is approximated using the Gaussian center. Consequently, strictly combining Gaussian rendering for both modalities leads to suboptimal image and LiDAR outputs. The proposed LiDAR visibility rate γ avoids this side effect while demonstrating improved novel view rendering with LiDAR assistance (Table 1).

5. Conclusions

In summary, we introduce a novel differentiable LiDAR rendering model for GS, enabling both LiDAR supervision and synthesis. Our results demonstrate that LiHi-GS achieves SOTA performance in novel view synthesis. We highlight the importance of LiDAR supervision in learning accurate scene geometry, which significantly enhances image rendering quality, especially under actor or ego shifts. Unlike prior works that primarily focus on urban datasets with dense sensor coverage, our approach is the first to evaluate on monotonous highway data, bridging the gap between existing research and real-world autonomous driving applications.

Limitation and improvements: LiHi-GS does not support deformable objects and harsh weather conditions. We also notice some non-flat Gaussians can cause geometry distortion on the ground; inspired by [11, 17, 19], we plan to make Gaussian flat to further improve rendering quality.

References

[1] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. *arXiv preprint arXiv:2404.06109*, 2024. 7

[2] Alexandra Carlson, Manikandasriram S Ramanagopal, Nathan Tseng, Matthew Johnson-Roberson, Ram Vasudevan, and Katherine A Skinner. Cloner: Camera-lidar fusion for occupancy grid-aided neural representations. *IEEE Robotics and Automation Letters*, 8(5):2812–2819, 2023. 2

[3] Qifeng Chen, Sheng Yang, Sicong Du, Tao Tang, Peng Chen, and Yuchi Huo. Lidar-gs: Real-time lidar re-simulation using gaussian splatting. *arXiv preprint arXiv:2410.05111*, 2024. 3

[4] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint arXiv:2311.18561*, 2023. 3, 2

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In *CVPR Workshop on the Future of Datasets in Vision*, page 1, 2015. 1

[6] Jiadi Cui, Junming Cao, Yuhui Zhong, Liao Wang, Fuqiang Zhao, Penghao Wang, Yifan Chen, Zhipeng He, Lan Xu, Yujiao Shi, et al. Letsgo: Large-scale garage modeling and rendering via lidar-assisted gaussian primitives. *arXiv preprint arXiv:2404.09748*, 2024. 2, 3, 4, 6

[7] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2

[9] Zhirui Gao, Renjiao Yi, Chenyang Zhu, Ke Zhuang, Wei Chen, and Kai Xu. Generic objects as pose probes for few-shot view synthesis. *arXiv preprint arXiv:2408.16690*, 2024. 7

[10] Sheng Hong, Junjie He, Xinhu Zheng, Chunran Zheng, and Shaojie Shen. Liv-gaussmap: Lidar-inertial-visual fusion for real-time 3d radiance field map rendering. *IEEE Robotics and Automation Letters*, 2024. 2, 3

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 8, 3

[12] Nan Huang, Xiaobao Wei, Wenzhao Zheng, Pengju An, Ming Lu, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Shanghang Zhang. S3 gaussian: Self-supervised street gaussians for autonomous driving. *arXiv preprint arXiv:2405.20323*, 2024. 2, 3, 4, 6

[13] Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18236–18246, 2023. 2

[14] Seth Isaacson, Pou-Chun Kung, Mani Ramanagopal, Ram Vasudevan, and Katherine A Skinner. Loner: Lidar only neural representations for real-time slam. *IEEE Robotics and Automation Letters*, 2023. 2

[15] Changjian Jiang, Ruilan Gao, Kele Shao, Yue Wang, Rong Xiong, and Yu Zhang. Li-gs: Gaussian splatting with lidar incorporated for accurate large-scale reconstruction. *arXiv preprint arXiv:2409.12899*, 2024. 3

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2, 3, 6, 1

[17] Mustafa Khan, Hamidreza Fazlali, Dhruv Sharma, Tongtong Cao, Dongfeng Bai, Yuan Ren, and Bingbing Liu. Autosplat: Constrained gaussian splatting for autonomous driving scene reconstruction. *arXiv preprint arXiv:2407.02598*, 2024. 2, 3, 4, 6, 8

[18] Akshay Krishnan, Amit Raj, Xianling Zhang, Alexandra Carlson, Nathan Tseng, Sandhya Sridhar, Nikita Jaipuria, and James Hays. Lane: Lighting-aware neural fields for compositional scene synthesis. *arXiv preprint arXiv:2304.03280*, 2023. 2

[19] Pou-Chun Kung, Seth Isaacson, Ram Vasudevan, and Katherine A Skinner. Sad-gs: Shape-aligned depth-supervised gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2842–2851, 2024. 8, 3

[20] Mingrui Li, Jingwei Huang, Lei Sun, Aaron Xuxiang Tian, Tianchen Deng, and Hongyu Wang. Ngm-slam: Gaussian splatting slam with radiance field submap. *arXiv preprint arXiv:2405.05702*, 2024. 3

[21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 6

[23] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 2, 4

[24] Betsy Reed. Tesla autopilot feature was involved in 13 fatal crashes, us regulator says, 2024. Accessed: 2024-04-26. 1

[25] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 2

[26] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 2

[27] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 6

[28] Tang Tao, Longfei Gao, Guangrun Wang, Yixing Lao, Peng Chen, Hengshuang Zhao, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 390–398, 2024. 2

[29] 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*, pages 14895–14904, 2024. 1, 2, 3, 5, 6, 7

[30] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12931, 2022. 2

[31] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12375–12385, 2023. 2

[32] Hanfeng Wu, Xingxing Zuo, Stefan Leutenegger, Or Litany, Konrad Schindler, and Shengyu Huang. Dynamic lidar resimulation using compositional neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19988–19998, 2024. 2

[33] Zirui Wu, Tianyu Liu, Liyi Luo, Zhide Zhong, Jianteng Chen, Hongmin Xiao, Chao Hou, Haozhe Lou, Yuantao Chen, Runyi Yang, et al. Mars: An instance-aware, modular and realistic simulator for autonomous driving. In *CAAI International Conference on Artificial Intelligence*, pages 3–15. Springer, 2023. 2

[34] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 6

[35] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024. 1, 2, 3, 4, 5, 6, 7

[36] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2

[37] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023. 2

[38] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 6

[39] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 3

[40] Xianling Zhang, Nathan Tseng, Ameerah Syed, Rohan Bhasin, and Nikita Jaipuria. Simbar: Single image-based scene relighting for effective data augmentation for automated driving vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3718–3728, 2022. 2

[41] Cheng Zhao, Su Sun, Ruoyu Wang, Yuliang Guo, Jun-Jun Wan, Zhou Huang, Xinyu Huang, Yingjie Victor Chen, and Liu Ren. Tclc-gs: Tightly coupled lidar-camera gaussian splatting for surrounding autonomous driving scenes. *arXiv preprint arXiv:2404.02410*, 2024. 2, 3

[42] Zehan Zheng, Fan Lu, Weiyi Xue, Guang Chen, and Changjun Jiang. Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5145–5154, 2024. 2

[43] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21336–21345, 2024. 2, 3, 4

[44] 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*, pages 21634–21643, 2024. 2, 3, 4

LiHi-GS: LiDAR-Supervised Gaussian Splatting for Highway Driving Scene Reconstruction

Supplementary Material

A. Baselines Implementation

Here are detailed descriptions of our baseline implementations. We unified the train/test frame selection strategy to make sure all the methods are using the same training and testing frames.

Instant-NGP. We use the Nerfstudio implementation of instant-NGP with default configuration.

3DGS. We use Splatfacto-big, a more efficient implementation from Nerfstudio, which has been shown to outperform the official version. To integrate LiDAR measurements, we initialize the Gaussian using a voxel-downsampled LiDAR point cloud map combined with COLMAP sparse points.

NeuRAD. We utilize the official implementation with the NeuRAD-paper configuration to ensure a direct comparison with the settings proposed in the paper.

StreetGS. We use the Nerfstudio implementation of StreetGS as our baseline. While this implementation has been shown to deliver similar or better rendering quality compared to the original, we incorporate several missing features to ensure a fair comparison with StreetGS. First, we added the sampling and pruning steps, as described in the implementation details of [35], to prevent object Gaussians from extending beyond the bounding box. Second, we implemented pose optimization for bounding box refinement. Third, we applied depth image loss following the original implementation. To mitigate the impact of noisy LiDAR observations, we optimize only the 95% of pixels with the smallest depth error. Additionally, we set λ_{depth} to 0.01 following [35].

B. Losses Details

The total optimization objective is as follows:

$$\mathcal{L} = \mathcal{L}_c + \lambda_1 \mathcal{L}_{lidar} + \lambda_2 \mathcal{L}_{opa}^L + \lambda_3 \mathcal{L}_{reg}^s + \lambda_4 \mathcal{L}_{opa}^c + \lambda_5 \mathcal{L}_{reg}^{pose} \tag{25}$$

$$\mathcal{L}_{opa}^c = \mathcal{L}_{sky} + \mathcal{L}_{fg} + \mathcal{L}_{reg}^{obj} \tag{26}$$

RGB Loss. The color loss $\mathcal{L}_c$ is designed to minimize the difference between color in the image, C_{img}, and rendered color, $\hat{C}$. We use the combination of L1 loss and D-SSIM loss following [16]:

$$\mathcal{L}_{color} = (1 - \lambda_{ssim})\mathcal{L}_1 + \lambda_{ssim}\mathcal{L}_{D-SSIM} \tag{27}$$

$$\mathcal{L}_1 = \|\mathcal{C}_{img} - \hat{\mathcal{C}}\|_1 \tag{28}$$

and λ_{ssim} is the weight of D-SSIM loss. We use $\lambda_{ssim} = 0.2$ in our experiment.

LiDAR Loss. Our LiDAR loss $\mathcal{L}_{lidar}$ is computed by the L1 difference between the range image, $\mathcal{R}_{img}$, and rendered range image, $\hat{\mathcal{R}}$:

$$\mathcal{L}_{lidar} = \|\mathcal{R}_{img} - \hat{\mathcal{R}}\|_1 \tag{29}$$

The range image pixel with Nan value is ignored.

LiDAR Accumulation Loss. Since LiDAR has 360 field of view that includes some areas without camera coverage, we add an extra loss $\mathcal{L}_{opa}^L$ to encourage the LiDAR visibility $\alpha_{lidar} = 1$ for the pixel with depth return in the range image. The qualitative result of LiDAR accumulation loss is shown in 12

Sky Loss. $\mathcal{L}_{sky}$ is guiding sky pixels in an image to have $\alpha = 0$. This makes sure the sky pixels are rendered from the sky model.

Foreground Loss. $\mathcal{L}_{fg}$ is guiding foreground pixels in an image to have $\alpha = 1$. We choose semantic labels in "flat," "human," "vehicle," "construction," and "object" groups as foreground in Cityscapes [5] class definitions.

Note that [35] propose using binary cross-entropy loss that classifies the image to the sky and non-sky region, which works well in urban scenes with clear foreground-background separation. However, distant elements like fields, mountains, or cityscapes should also be considered background in our open highway scenes. Our proposed loss promotes low opacity for sky pixels and high opacity for specific foreground segments, offering the model greater flexibility in distinguishing foreground from background.

Scale Reg Loss. $\mathcal{L}_{reg}^s$ is a scale regularization term to reduce spike-like Gaussian to produce more accurate depth rendering and avoid spike artifacts when novel view rendering. We set the maximum anisotropy ratio to 3. We notice this loss term has a great impact on novel view rendering quality.

Pose Reg Loss. $\mathcal{L}_{reg}^{pose}$ is a pose regularization term to make sure the optimized pose T_{opt} is not drifting too far away from the original labeled pose T.

$$T = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{30}$$

$$\Delta\mathbf{t} = \mathbf{t}_{opt} - \mathbf{t} \tag{31}$$

$$\Delta\mathbf{R} = \mathbf{R}_{opt}^\top \mathbf{R} \tag{32}$$

$$\Delta\theta = \cos^{-1}\left(\frac{\text{trace}(\Delta\mathbf{R}) - 1}{2}\right) \tag{33}$$

$$\mathcal{L}_{reg}^{pose} = \lambda_{trans}\Delta\mathbf{t} + \lambda_{rot}\Delta\theta \tag{34}$$

Figure 12. The rendered LiDAR range image comparison with and without LiDAR opacity loss. The results without LiDAR opacity loss include holes in the areas without camera coverage.

In practice, we set $\lambda_{trans} = 1$ and $\lambda_{rot} = 100$.

Depth Image Loss. The depth image loss use in [35] can be computed by the L1 difference between the depth image, D_{img}, and rendered depth, $\hat{D}$:

$$\mathcal{L}_{depth} = \|D_{img} - \hat{D}\|_1 \tag{35}$$

This loss is not used in proposed LiHi-GS and is only used to compare the difference between our proposed LiDAR modeling and the depth image loss used in prior works.

C. Experiments Details

In all of our experiment, we set loss parameters $\lambda_1 = 0.5, \lambda_2 = 0.1, \lambda_3 = 10, \lambda_4 = 0.5, \lambda_5 = 0.01$. For the densification step, the densification starts after 3000 iterations with 100 iterations interval, and it stops after 25000 iterations. The gradient threshold is 0.0004. The opacity is reset every 3000 iteration. The total iteration number is set to 50000 iterations.

Preprocessing. We generate the Structure-from-Motion (SfM) point cloud using COLMAP with pre-defined camera poses. Following the approach in [35], we exclude moving objects by applying a mask. The LiDAR point cloud map is created by aggregating individual LiDAR scans and applying a 2-meter voxel-downsample filter. The initial Gaussian means are derived from both the COLMAP points and the LiDAR map. For constructing the actors' point cloud, we aggregate the LiDAR points within the labeled bounding

Figure 13. Illustration of the effect of 2D scale compensation.

	PSNR	LiDAR L1 Med.	Memory
W/ Scale Comp.	31.63	0.63	37Gb
W/O Scale Comp.	30.22	0.64	> 80Gb

Table 4. Scale Compensation Ablation Studies.

boxes and apply a 0.2-meter voxel-downsample filter to the actor point cloud, reducing memory consumption.

Background Model. We follow [4] to model long-range background objects and the sky with a learnable environment cube map. The cube map outputs a view-dependent background image C_{bg}. With the rendered color $\hat{C}(p)$ and rendered opacity $O(p)$ at pixel p, the final rendered color can be compute as:

$$C(p) = \hat{C}(p) + (1 - O(p)) \cdot C_{bg}(p) \tag{36}$$

where the rendered opacity is:

$$O(p) = \sum_{i \in N} \alpha_i' \prod_{j=1}^{i-1} (1 - \alpha_j') \tag{37}$$

D. 2D Gaussian Scale Compensation Details

2D Gaussian scale compensation is aimed at avoiding small Gaussians that are lost during projection when rendering Li-DAR range images. Figure 13 shows the core idea of this scale compensation. Disabling this step can cause memory overhead due to the expanded Gaussian with wrong geometry at long range. The ablation studies with memory usage are shown in Table 4

E. Qualitative Rendering Results

In this section, we demonstrate the results of a qualitative comparison with our main competitors (StreetGS and Neu-RAD). Figure 14 and Figure 15 demonstrate LiHi-GS provides overall the best rendering quality on both static and

Figure 14. Comparison of dynamic object reconstructions at varying distances from the ego vehicle (13m, 22m, 31m, and 52m). The first three columns show rear-view reconstructions, while the last column shows a front-view reconstruction. LiHi-GS (bottom row) demonstrates superior reconstruction quality across all cases: In column 1, it accurately captures both the color and headlights of the blue SUV, while NeuRAD shows color inaccuracies and StreetGS exhibits blurriness. In column 3, LiHi-GS preserves fine details of the cargo trailer's fronthood structure that are lost in both baseline methods. In column 4, LiHi-GS achieves complete vehicle reconstruction with clear details down to the license plate area, while StreetGS produces incomplete geometry and NeuRAD shows significant blurring.

dynamic objects across different distances. Figure 16 show LiHi-GS offers better dynamic and static scene decomposition than StreetGS. StreetGS fails to model some moving actors in the scene, while proposed LiHi-GS successfully models all moving actors with proposed LiDAR supervision and camera-LiDAR decoupled pose optimization.

F. Limitations

LiHi-GS has some known limitations. First, the method is limited to reconstructing moving objects with rigid body motion. We are not able to model non-rigid objects like pedestrians or animals.

Secondly, we noticed some limitations in our GS-based method when performing extreme actor shifting, as shown in Figure 17. In this scenario, the actor is only partially observed; while both StreetGS and our method fail to reconstruct the missing part of the vehicle, NeuRAD demonstrates the ability to predict the remaining geometry and achieve better completion. However, the texture and color of the vehicle are still noisy in NeuRAD's reconstruction. To improve actor completion in our method, we could leverage the reflected Gaussian consistency proposed in[17],

which addresses partially missing observations by assuming symmetry in the vehicle's structure. Additionally, the regions originally occluded by actors exhibit greater noise in GS-based methods than NeuRAD. This is likely also due to NeRF-based methods offering better interpolation and hole-filling capabilities, producing plausible results in unseen areas where GS-based methods have difficulties. A potential solution is to hybrid NeRF and GS-based method for background scene as proposed in [20]

Thirdly, as shown in Table 1 of the main paper, our method achieves comparable or slightly higher LiDAR L1 median values than NeuRAD. We believe this is primarily due to NeuRAD's more accurate ray tracing depth rendering compared to our current projection-based approach. Specifically, our method approximates the depth of a Gaussian using the depth of its center, which may result in less precise depth estimation than NeRF-based methods. However, recent advancements in rendering accurate depth from GS, such as those proposed in [7, 11, 19, 39], offer promising solutions to this limitation. We mark this as a future work to improve the depth estimation.

Figure 15. Comparison of static object reconstructions at varying distances from the ego vehicle (5m, 18m, and 53m). The first two columns show rear-view reconstructions, while the last column shows a front-view reconstruction. LiHi-GS (bottom row) presents overall best rendering quality in all static objectcases: In column 1, it achieves precise lane line reconstruction with clear geometry, while StreetGS shows significant blurring and NeuRAD exhibits geometric distortions in the lane markings. In column 2, LiHi-GS preserves both the shape and color correctness of the red sign, which appear degraded in both baseline methods. In column 3, LiHi-GS maintains legible text on a matrix board at 53m distance, while NeuRAD and StreetGS fail to preserve text clarity at this range.

Figure 16. Actor only and actor removal. The left sequence is captured from the front camera, and the right sequence is from the rear-view camera. LiHi-GS shows better static and moving scene decomposition than StreetGS. StreetGS fails to model some actors due to wrong geometry and camera-LiDAR bounding box misalignment. With the proposed LiDAR supervision and decoupled pose optimization, LiHi-GS performs better scene decomposition.

Figure 17. Failure case when performing extreme lateral actor shifting. The orange box highlights the reconstruction of a partially occluded actor. GS-based methods fail to reconstruct the occluded region. While NeuRAD provides better completion, the reconstruction is noisy. The red box indicates that the NeRF-based method offers better hole-filling capability than GS-based methods in unseen regions.