Hybrid Rendering for Multimodal Autonomous Driving: Merging Neural and Physics-Based Simulation

Máté Tóth Péter Kovács Réka Bencses Zoltán Bendefy Zoltán Hortsin Balázs Teréki Tamás Matuszka

aimotive.com

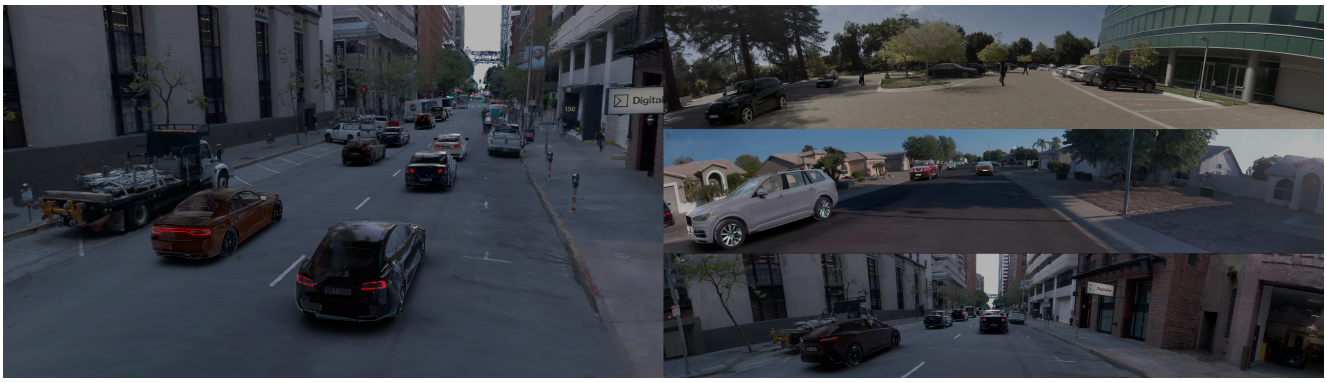


Figure 0. We introduce a novel neural reconstruction and hybrid rendering pipeline designed to generate high-fidelity 3D reconstructions of static environments. Our approach enables the seamless integration of dynamic actors and ensures smooth transitions between mesh-based and Gaussian Splatting representations, facilitating multimodal sensor simulation.

Abstract

Neural reconstruction models for autonomous driving simulation have made significant strides in recent years, with dynamic models becoming increasingly prevalent. However, these models are typically limited to handling in-domain objects closely following their original trajectories. We introduce a hybrid approach that combines the strengths of neural reconstruction with physics-based rendering. This method enables the virtual placement of traditional mesh-based dynamic agents at arbitrary locations, adjustments to environmental conditions, and rendering from novel camera viewpoints. Our approach significantly enhances novel view synthesis quality, especially for road surfaces and lane markings, while maintaining interactive frame rates through our novel training method, NeRF2GS. This technique leverages the superior generalization capabilities of NeRF-based methods and the real-time rendering speed of 3D Gaussian Splatting (3DGS). We achieve this by training a customized NeRF model on the original images with depth regularization derived from a noisy LiDAR point cloud, then using it as a teacher model for 3DGS training. This process ensures accurate depth, surface normals, and camera appearance modeling as supervision. With our block-based training parallelization, the method can handle large-scale reconstructions ($\geq$ 100000 m^2) and predict segmentation masks, surface normals, and depth maps. During simulation, it supports a rasterization-based rendering backend with depth-based composition and multiple camera models for real-time camera simulation, as well as a ray-traced backend for precise LiDAR simulation.

1. Introduction

The development of robust autonomous driving systems depends heavily on diverse datasets for training and evaluation. Traditional approaches rely on recordings of real-world driving scenarios. However, datasets such as [2, 3, 16, 24] often lack critical safety-related edge cases due to their rarity. Capturing these infrequent events is both costly and logistically challenging, making alternative approaches necessary. As a result, there is growing interest in synthetic data generation techniques, which offer a promising solution to bridge the gap in safety-critical scenarios for autonomous vehicle development.

Traditional simulation methods for autonomous driving, such as CARLA [6], are based on principles similar to those used in 3D game engines. They rely on mesh-based 3D models and traditional computer graphics rendering techniques, such as rasterization or ray tracing. While these simulators provide a high degree of control over the sim-

ulated environment, they have two major drawbacks. First, creating detailed 3D environments and assets requires extensive manual effort. Second, an unavoidable domain gap remains between simulated and real-world data.

Neural Radiance Fields (NeRF) [17] significantly improved reconstruction quality compared to earlier data-driven 3D scene reconstruction methods. More recently, 3D Gaussian Splatting (3DGS) [12] introduced an explicit scene representation that enables real-time rendering while maintaining near-photorealistic reconstruction quality. These advancements have made data-driven neural reconstruction methods a viable option for automotive simulation.

Several adaptations have been made to 3DGS for automotive applications, primarily focusing on reconstructing dynamic scenes using supervised and unsupervised settings and integrating additional sensor modalities used in autonomous driving, such as LiDAR. However, these methods still lack one of the key advantages of traditional simulation techniques: controllability. Inserting unseen objects or repositioning objects far from their original trajectory remains challenging, often leading to degraded reconstruction quality and limiting their effectiveness for autonomous driving downstream tasks.

In this paper, we present a hybrid rendering method that combines the strengths of neural reconstruction and traditional computer graphics to overcome their respective limitations. Our approach reconstructs the static environment using neural rendering while incorporating dynamic actors through conventional graphics techniques. This enables flexible placement of arbitrary objects, adjustable weather conditions, and rendering from novel viewpoints. By leveraging neural rendering, we reduce the domain gap commonly associated with traditional simulators while preserving their controllability. This allows us to mitigate the weaknesses of both methods. Additionally, our solution supports multimodal output, including camera sensor simulation, LiDAR point clouds, depth maps, and surface normals. To demonstrate the effectiveness of our approach, we conduct experiments on data generated by our hybrid method, evaluating its suitability for autonomous driving development. In summary, our main contributions are the following:

- A hybrid rendering approach that combines neural reconstruction and traditional computer graphics techniques to overcome their limitations.
- Enhanced novel view synthesis introducing NeRF2GS method.
- Benchmark results for downstream autonomous driving tasks using data generated by our method.

2. Related Work

Many efforts have been made to adapt neural reconstruction methods for automotive simulation. One of the key challenges in scene reconstruction is dealing with dynamic objects. Current approaches address this issue by attempting to reconstruct the original dynamic objects in the scene using either supervised or unsupervised learning techniques. Supervised NeRF-based methods, such as Neural Scene Graphs [20], MARS [32], and NeuRAD [27], use bounding box data to learn how to reconstruct dynamic objects as static ones in a canonical space, and then transform them back to their original positions in world space. On the other hand, the unsupervised EmerNeRF [34] method focuses on reconstructing dynamic objects using a temporally deformable dynamic field. While some of these methods achieve impressive reconstruction quality, they are not suitable for interactive simulator applications due to the slow rendering speeds of NeRFs.

This limitation is largely addressed by 3DGS-based reconstruction methods such as the unsupervised S3Gaussian [8], Periodic Vibration Gaussian [4], and DeSiRe-GS [21], which achieve high-quality reconstructions of dynamic scenes along the training trajectories. However, their ability to manipulate dynamic actors is severely restricted, and they only support RGB camera simulation, which is often insufficient for autonomous driving applications.

Supervised 3DGS methods, including Street Gaussians [33], DrivingGaussian [40], and OmniRe [5], provide some control over dynamic objects, allowing minor adjustments to their positions and rotations, but it is still limited. Additionally, they do not support LiDAR simulation or arbitrary camera models. SplatAD [7] extends 3DGS-based methods by incorporating LiDAR simulation, but the other aforementioned limitations remain unresolved.

3. Hybrid Rendering for Autonomous Driving

Our proposed hybrid rendering method addresses the challenges mentioned in Section 2 by integrating neural reconstruction and a traditional computer graphics rendering pipeline, each composed of multiple specialized submodules. Figure 1 depicts the neural reconstruction model generation process, detailing the transformation from raw input to the final model while the high-level overview of the rendering pipeline is visualized in Figure 3.

3.1. Preparing training data for neural reconstruction

Our pipeline uses four input data sources for the reconstruction: *images* from the high-resolution onboard cameras of the vehicle, *point clouds* generated by one or more onboard LiDARs, *egomotion* and *extrinsic and intrinsic calibration* of the cameras.

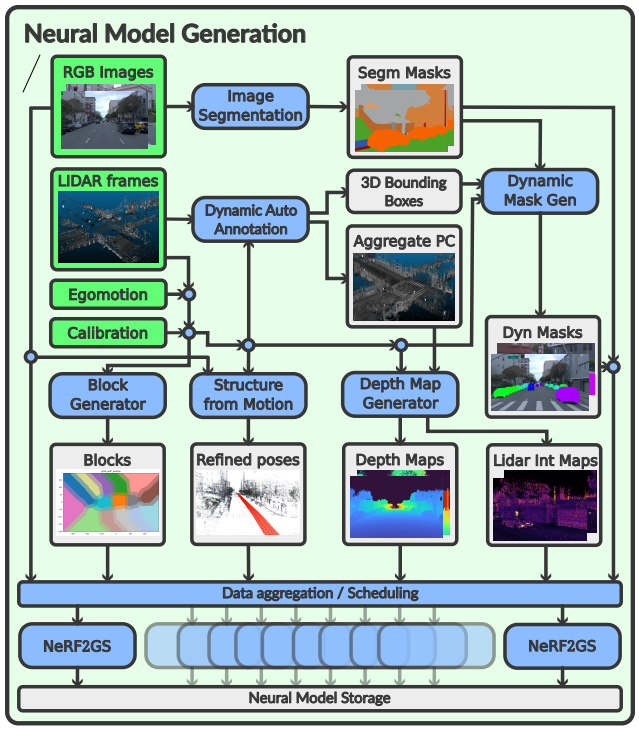


Figure 1. Overview of the reconstruction model generation method.

3.1.1. Dynamic object handling

Our method focuses on highly controllable reconstruction with strong extrapolation capabilities. Since current state-of-the-art approaches can only reconstruct dynamic scenes to a limited extent, we restrict our reconstruction to the temporally consistent, static parts of the scene. To achieve this, we mask out dynamic objects, such as pedestrians and moving vehicles, from the ground truth (GT) images.

First, we use OneFormer [9] to generate panoptic segmentations for the GT images. We then extract instance masks for relevant objects and track them in image space using an optical flow-based frame-to-frame tracking method. To identify potential dynamic objects, we use a 3D object detection model that processes LiDAR and camera inputs to generate 3D bounding boxes and determine their stationarity in world space. Next, we project LiDAR points from the current frame onto the instance masks, labeling them accordingly. By matching the bounding boxes with instance masks using the labels of enclosed LiDAR points, we effectively track segmentation masks while incorporating stationarity information. Finally, we refine and merge the image-space and bounding box-based tracklets, and create the set of dynamic masks by adding the instance masks corresponding to nonstationary tracklets.

3.1.2. LiDAR intensity and depth map generation

Neural reconstruction methods can generate near-photorealistic RGB renderings, but they often struggle to reconstruct accurate geometry in cases of sparse training views or weakly textured and blurred regions, such as road surfaces. Since our method aims to simulate LiDAR sensors, depth supervision and LiDAR intensity rendering are essential.

To achieve this, we generate dense depth and intensity maps from LiDAR data. We first create a static aggregated point cloud by removing LiDAR points that fall within the bounding boxes of dynamic objects in each sweep, Then merge the sweeps using per-point egomotion correction. Then we split the aggregated point cloud into square chunks in BEV space, and employ Poisson surface reconstruction [11] for each chunk to generate a dense, mesh-based representation of the lidar data. Finally, we use Open3D [39] to render the dense depth maps and LiDAR intensity images from these meshes using the intrinsic and extrinsic camera parameters of the train camera frames.

3.1.3. Camera pose refinement

While high-fidelity 3D scene reconstruction requires highly accurate intrinsic calibration and camera poses, the egomotion-derived world-space camera extrinsic matrices in automotive datasets are often noisy. To address this, we refine camera poses and intrinsic parameters using a modified version of COLMAP [23]. Starting with the original calibration as initialization, we iteratively apply triangulation and bundle adjustment until convergence.

Since structure-from-motion methods are invariant to translation, rotation, and scale, we must transform the optimized poses back to the original world-space. To achieve this, we apply the Kabsch algorithm [10] between the original and optimized camera positions.

3.1.4. Block generation

Autonomous driving tasks often require large-scale scene reconstruction in a highly scalable manner. However, neural reconstruction methods do not scale well to very large scenes. There are several approaches, like Mega-NeRF [28], SUDS [29], and Block-NeRF [25] that address this issue using a square grid or clustering-based space partitioning approach, by training separate reconstruction model for each of the blocks. Our solution is motivated by [29] and it is also based on clustering. We decompose the scene into multiple overlapping blocks using a dynamic bird's-eye view (BEV) raster-based block representation.

First, we transform the camera positions into a 2D BEV space and iteratively apply k-nearest neighbors clustering, gradually increasing the cluster count in each iteration until the predefined maximum cluster radius and image count conditions are met. We then discretize the BEV space into a 2D raster (where each cell can be contained by multiple

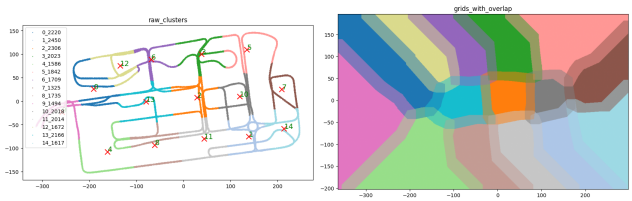


Figure 2. Clustered BEV training poses (left) and the generated blocks with overlaps (right)

blocks at the same time), and assign each cell to the cluster with the highest number of points. Next, we expand the clusters to their convex hulls and assign any remaining unassigned cells to the nearest cluster. Finally, we apply dilation to the clusters to ensure a specified level of overlap.

This approach guarantees overlap between blocks, which is crucial for minimizing popping artifacts at block edges while remaining flexible enough to efficiently cover irregular camera trajectories. It also makes interpolation possible between the distinct reconstruction models based on the cluster edge distances, which greatly reduces artifacts in case of traversing through block boundaries. Figure 2 shows the output of the block generation algorithm on a complex scenario.

3.2. NeRF2GS reconstruction method

Explicit, point-based methods like 3DGS provide reconstruction quality comparable to state-of-the-art implicit methods such as Zip-NeRF [1], with the added benefit of real-time rendering capabilities. However, they still fall short in terms of extrapolation, particularly when trained with sparse input views or under varying lighting conditions. Research has shown that training an implicit model (such as NeRF) on the original data and using it as both initialization and supervision for training a Gaussian-splatting-based model can significantly enhance Novel View Synthesis quality [19]. Our reconstruction method, NeRF2GS, builds on this approach: we first train a customized NeRF model on the original RGB images, supervised by dense LiDAR depth maps, and incorporate a learned camera exposure correction method. We then render the outputs of this model at the original training poses and generate a point cloud, which serves as ground truth data and provides initialization for training a 3DGS-based reconstruction.

3.2.1. NeRF training

We use a modified version of the Nerfacto model within the Nerfstudio framework [26], incorporating several improvements to enhance reconstruction quality. Depth supervision is implemented using the depth loss proposed in Urban Radiance Fields [22] with segmentation-based loss weighting and an exponentially decaying loss scheduling scheme. To mitigate floating artifacts in sky regions where no depth ground truth is available, we apply L2 regularization to the density accumulation in areas classified as sky in the ground truth segmentation.

Additionally, we introduce a LiDAR intensity prediction network—a multi-layer perceptron (MLP) that takes the geometric feature vector from the Nerfacto density field, concatenated with a sinusoidal position encoding, and predicts an intensity value. This prediction is supervised using dense LiDAR intensity maps with an L2 loss. We also incorporate learned segmentation in a similar manner to [38]. We predict a 32-dimensional vector representing raw logits corresponding to segmentation classes, and train it using a categorical cross-entropy loss on the semantic segmentation maps.

Automotive recordings typically use cameras with automatically adjusted ISP (Image Signal Processing) parameters, such as auto gain, auto white balance, and local tone mapping, which can introduce inconsistencies in reconstruction. To address this, the common approach is to use some form of appearance embedding [15]. However, we opted for the bilateral guided ISP disentanglement method proposed in [31], which better preserves image consistency. During training, we optimize a 3D Bilateral Grid for each training image as described in [31], effectively creating an "ISP enhancement-free" scene representation. However, when rendering images for 3DGS training, we omit the post-processing step, as ISP adjustments are handled by the hybrid rendering engine.

3.2.2. 3D Gaussian splatting training

Our Gaussian Splatting methodology builds upon the established training approach introduced in [12], incorporating the enhanced AbsGS densification strategy [36] and utilizing gsplat[35] as its backend. We employ direct depth and normal regularization to improve model robustness in novel view synthesis and provide these attributes for subsequent stages within our hybrid rendering framework. Additionally, LiDAR intensity and segmentation classes are embedded as additional per-Gaussian features. A preceding NeRF model is leveraged to render color-corrected RGB, depth, normal, LiDAR intensity, and segmentation images, and to augment the initial COLMAP point cloud with a dense point cloud for 3DGS initialization.

The normal regularization is inspired by the method of DN-Splatter [30], wherein the normal vector of each Gaussian is defined as the directional vector corresponding to its minimum scale component:

$$\hat{\mathbf{n}}_\mathbf{i} = \mathbf{R}_\mathbf{i} \cdot \text{OneHot}(\arg\min\,(s_{i,1}, s_{i,2}, s_{i,3})) \qquad (1)$$

where $\mathbf{R}_\mathbf{i}$ represents the rotation matrix of the i^{th} Gaussian, and s_i denotes the associated scale vector. These computed normal vectors are then rendered analogously to color chan-

nels, and the corresponding loss term is formulated as:

$$\mathcal{L}_{norm} = -\lambda_{norm} \hat{\mathbf{n}}_{\mathbf{pred}} \cdot \hat{\mathbf{n}}_{\mathbf{gt}} \tag{2}$$

To ensure the well-definedness of the normal vectors, we additionally employ a regularization term that encourages the Gaussians to contract along their minimum scale dimension until a predetermined size threshold is reached:

$$\mathcal{L}_{flat} = \sum_{i} \text{ReLU}(\min(s_{i,1}, s_{i,2}, s_{i,3}) - C_{flat}) \tag{3}$$

LiDAR intensity values are incorporated as supplementary features for each Gaussian. These features are normalized to the range [0, 1] via a sigmoid activation function and subsequently rendered as color channels. An L1 loss function is employed to train these features, aiming to reproduce the dense reflectivity images generated by the NeRF model:

$$\mathcal{L}_{lidar} = \lambda_{lidar} |I_{pred} - I_{gt}| \tag{4}$$

Segmentation labels are trained analogously to the LiDAR intensities. Our internal 32 segmentation labels are transformed into 6-bit binary representations, and the corresponding six additional Gaussian features are trained independently using the same L1 loss as defined in Equation 4, to replicate these binary values:

$$\mathcal{L}_{segm} = \sum_{i=0}^{6} |S_{pred,i} - S_{gt,i}| \tag{5}$$

where S_i represents the image rendered from the i$^{\text{th}}$ segmentation feature. During inference, each of the six rendered segmentation bits is rounded to the nearest integer, and the segmentation label ID is reconstructed by interpreting these rounded values as the binary digits of the label identifier.

To effectively represent far-field regions, we decompose the scene into two distinct sets: ground Gaussians and sky Gaussians. While the ground Gaussians represent the local 3D structure, the sky Gaussians are constrained to the surface of a distant sphere. During rendering, the final pixel color is determined by blending the contributions of both models based on the accumulated opacity of the ground Gaussians:

$$C_{final} = C_{ground} + (1 - A_{ground}) C_{sky} \tag{6}$$

where A_{ground} represents the total accumulated alpha from the ground model.

To enforce a clean separation between these layers, we leverage the segmentation masks M_{sky} derived from the preceding NeRF model. We introduce an accumulation loss, $\mathcal{L}_{acc}$, which incentivizes the ground model to reach full opacity in non-sky regions and zero opacity in sky regions:

$$\mathcal{L}_{acc} = \lambda_{acc} \| A_{ground} - (1 - M_{sky}) \| \tag{7}$$

This constraint ensures that the sky Gaussians are only visible in areas explicitly labeled as sky, preventing spatial overlap and geometric ambiguity between the two representations.

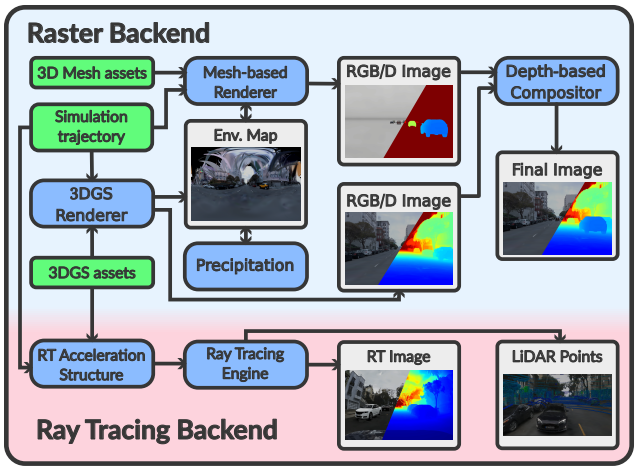


Figure 3. Overview of the rendering pipeline used in our method.

3.3. Rendering pipeline

The original rendering algorithms [12, 41] are designed for perspective projection. Our goal was to generalize the evaluation of 3D Gaussians adapting it to arbitrary sensors; therefore, we needed to avoid any kind of projection. The result is a small, high performance, easily portable algorithm utilizing the response calculation method based on [18]. Portability lets us use a shared codebase for rasterization and ray tracing for calculating the contribution of a Gaussian along a ray. This solution also allows us to mitigate popping artifacts and expand the possibilities of neural reconstruction, e.g. with LiDAR simulation.

We integrated this method into our rendering systems, one based on rasterization and one based on ray tracing, where both techniques support arbitrary camera distortion models (fisheye, equirectangular, etc.).

3.3.1. Rasterization

Wide-angle or equirectangular camera distortion models typically require 360-degree information, which is rendered using six virtual perspective cameras. These camera outputs must be seamlessly stitched, a task that traditional algorithms struggle with. However, our solution bypasses this challenge.

In our approach, during the rasterization of the Gaussians' 3D proxy geometry, we calculate rays for each pixel

Figure 4. Qualitative comparison of novel view synthesis results between OmniRe (left) and our model (right) for the static components of the scene. Differences in reconstruction quality are particularly noticeable in the rendering of lane boundaries and road markings.

and evaluate their contribution based on the maximum response along the ray. Unlike traditional methods that rely on projecting rectangles, our method focuses solely on rays intersecting the proxy geometry, ensuring consistent results across all directions and projections. This eliminates stitching artifacts, allowing us to avoid popping effects by sorting the data based on distance from the viewpoint rather than the z-distance from the near planes of the virtual cameras.

Since 3D Gaussians already provide depth information, the rasterization process can be continued by rendering dynamic mesh-based objects using traditional depth-testing methods. For shading these mesh objects, we use Image-based Lighting (IBL), which captures the environmental lighting and reflections. When a tessellated ground surface is available, we can render screen space ambient occlusion at the contact area between the mesh and the neural-rendered environment. Additionally, the depth compositing enables us to introduce weather effects, such as snow or rain, which are influenced by the IBL. Finally, the images from the virtual cameras are stitched together into a single image, based on the camera distortion model.

3.3.2. Ray tracing

The ray tracing algorithm employed for scene rendering follows the approach outlined in [18]. This method leverages the hardware-accelerated ray tracing pipeline, where each 3D Gaussian is represented as a proxy geometry within the acceleration structure. Rays are generated in the ray-generation shader, based on either the camera distortion model or a LiDAR scanning pattern. During the acceleration structure traversal, the splats along a ray are sorted by distance into a k-sized hit-buffer within the any-hit shaders. This hit-buffer is then processed in the ray-generation shader to compute the individual contributions of the intersected Gaussians, considering their maximum response along the ray. In addition to generating reflections, surface normal information encoded within the Gaussians is used to simulate the occlusion of ambient light incoming to the surface.

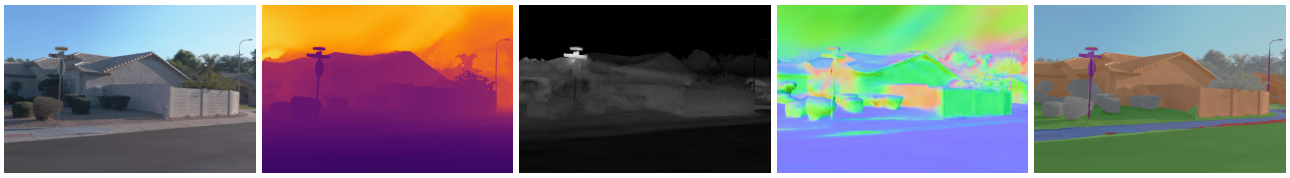
4. Results and Experiments

To qualitatively evaluate the effectiveness of our reconstruction method, we visualize its output across multiple modalities. Figure 6 demonstrates that our approach produces high-quality 3D reconstructions, accurately capturing geometric details and surface properties across RGB, depth, LiDAR intensity, normal, and segmentation modalities. Figure 5 shows an example of LiDAR point cloud visualization of the hybrid rendering method.

4.1. Static scene reconstruction quality

Due to the application of bilateral grid-based color correction in the NeRF step of our reconstruction method, the traditional PSNR metric does not provide a reliable measure of reconstruction quality. Since the SSIM and LPIPS

Figure 5. Visualization of the LiDAR point cloud rendered by our method.



(a) RGB reconstruction (b) Depth image (c) Lidar intensities (d) Normal map (e) Learned segmentation

Figure 6. Novel view multimodal renderings from our GS model.

metrics are more sensitive to structural similarity than simple pixel-wise color differences, they are more useful in our case. However, they also remain sensitive to inherent color mismatches between the original and reconstructed images caused by the ISP disentanglement step which is reflected in the evaluation results. In Table 1, we present these metrics on six Waymo scenes compared to vanilla 3DGS and TCLC-GS [37] methods. Beyond these perceptual metrics, we further evaluate our method's efficacy through downstream task performance, which aligns more closely with our objective of generating high-fidelity simulation environments for autonomous vehicle development.

4.2. Novel view synthesis quality

We evaluate the novel view synthesis performance of our method in comparison with OmniRe [5], which is considered the state-of-the-art approach for supervised dynamic scene reconstruction, and DeSiRe-GS [21], a state-of-the-art unsupervised dynamic scene reconstruction approach for urban environments. In the case of OmniRe and DeSiRe-GS, we used their official repositories for training and metrics calculations. For quantitative evaluation, all models were trained on the same scenes from the Waymo dataset using all five cameras, and a separate set of held-out images was used for metric computation. Because our method focuses on reconstructing only the static components of the environment, we report PSNR and SSIM scores computed exclusively over the static regions. Table 2 presents a comparison of our method with OmniRe and DeSiRe-GS across three dynamic Waymo scenes using these metrics. Across the evaluated scenarios, our approach matches or exceeds the performance of existing state-of-the-art neural reconstruction techniques.

Although evaluating "left out" images from the original dataset yields quantitative insights, this setup primarily measures the models' interpolation capabilities and does not adequately reflect their robustness to significant viewpoint shifts relative to the training trajectory (e.g., lane changes), which are common in autonomous driving scenarios. To address this limitation, we also perform qualitative novel view synthesis comparisons between OmniRe and our method under laterally shifted ego trajectories of up to 5 meters. Figures 4 and 7 demonstrate the qualitative

Sequence	Our 3DGS SSIM↑/LPIPS↓	Vanilla 3DGS* SSIM↑/LPIPS↓	TCLC-GS* SSIM↑/LPIPS↓
10247954...	0.88/0.26	0.84/0.25	0.89/0.16
10713922...	0.82/0.33	0.84/0.25	0.88/0.17
11037651...	0.75/0.58	0.67/0.44	0.71/0.42
13469905...	0.87/0.26	0.84/0.26	0.88/0.16
14333744...	0.85/0.28	0.85/0.25	0.87/0.21
14663356...	0.84/0.38	0.86/0.25	0.90/0.18

Table 1. Reconstruction quality comparison on Waymo scenes. *marked results are taken from the TLCL-GS [37] paper.

differences in highly extrapolated views, focusing on autonomous driving critical features like lane markings, traffic signs, and pedestrian crossings. We assume our model achieves superior quality in these scenarios due to the explicit normal regularization. This regularization constrains the flat splats to align with scene surfaces, leading to a more realistic reconstruction of the actual geometry, in addition to the accurate rendering of RGB images.

4.3. Downstream tasks

3D object detection. To validate the effectiveness of our solution for 3D object detection, we used DEVIANT [13], a publicly available monocular 3D object detection model trained on the Waymo Open Dataset. We conducted both quantitative and qualitative experiments. For the quantitative analysis, we rendered the scenes from a novel

Figure 7. Qualitative comparison of novel view synthesis quality between OmniRe (left) and our model (right).

Sequence	Our 3DGS PSNR↑/SSIM↑	OmniRe PSNR↑/SSIM↑	DeSiRe-GS PSNR↑/SSIM↑
13299463...	*26.15/0.8011*	26.07/0.7908	**26.94/0.8071**
17860546...	**30.13/0.9017**	*29.89/0.8894*	28.58/0.8499
30154365...	**28.36/0.8767**	*27.62/0.8433*	26.03/0.7908

Table 2. Novel view synthesis quality comparison on Waymo scenes. The highest-performing method is highlighted in **bold**, while the second-best result is indicated in *italics*.

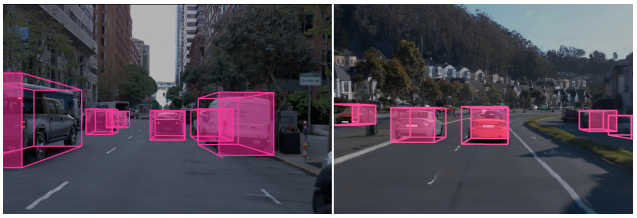


Figure 8. DEVIANT model detections on images rendered using our hybrid method.

$\mathbf{IoU}_{3D}$	Scenario	$\mathbf{AP}_{3D}$ [%] ↑			
		All	0-30	30-50	50-∞
0.3	Original	11.08	20.16	23.42	0.54
	Shifted (0 m)	9.40	23.88	13.73	0.39
	Shifted (1 m)	8.17	22.57	11.46	0.00
	Shifted (2 m)	9.17	22.38	15.04	0.00
	Shifted (3 m)	9.93	22.69	14.75	0.00

Table 3. Comparison of 3D object detection results on a scenario rendered from novel views.

$\mathbf{IoU}_{3D}$	$\mathbf{AP}_{3D}$ [%] ↑		$\mathbf{APH}_{3D}$ [%] ↑		Recall@0.95	
	0–30	30–50	0–30	30–50	0–30	30–50
0.5	100.00	52.94	99.05	52.09	100.00	0.00
0.7	66.67	40.00	65.92	39.87	0.00	0.00

Table 4. Comparison of 3D object detection results using the Detection Agreement metric.

viewpoint, shifting the ego-trajectory laterally by varying offsets (ranging from 0 to 3 meters) from *segment-13469905891836363794*, and used the transformed ground truth bounding boxes for metric calculation. The qualitative experiments involved running the DEVIANT model on three segments, where our reconstruction model rendered the static environment, and the mesh-based rendering engine introduced dynamic vehicles. Our primary focus was to assess whether simulator-generated vehicles introduce a domain gap.

Table 3 presents the detection results using Waymo detection metrics, where the 'Original' scenario refers to using the original images, while 'Shifted' scenarios are generated by our reconstruction model with laterally offset ego-trajectories. Interestingly, the model performs even better in novel view scenarios than on the original images at close range (0–30m), primarily due to a higher number of false positives in the original images. Distant object detection performance declines in novel view synthesis, warranting further investigation on a larger sample. However, detection results remain largely consistent across novel view scenarios. The observed differences can be attributed to object visibility changes caused by lateral shifts, as the model struggles to recognize truncated objects. In addition, we evaluated the detection results on the original and simulated images using the Detection Agreement metric [14] (see Table 4). The strong agreement between detections on original and simulated images suggests a small domain gap.

4.4. Performance metrics

Our system is fully capable of interactive rates on consumer hardware (e.g., NVIDIA RTX 3090). We achieve 12 ms ($\tilde{8}3$ FPS) for Full HD camera rendering and 17 ms ($\tilde{5}9$ FPS) for 64-beam LiDAR simulation. Furthermore, the pipeline utilizes multi-GPU distribution to maintain low latency in complex multi-sensor setups, making it robust for Hardware-in-the-Loop applications.

5. Limitations and Future Work

Limitations. The proposed block-based rendering approach, while designed to minimize discontinuities at block boundaries through sufficient overlap, can still produce visible artifacts, particularly in regions with highly textured distant objects. Additionally, the current representation of LiDAR intensities as static, view-independent values for each Gaussian does not accurately reflect the inherent directional dependence observed in real LiDAR sensors. While the evaluation of autonomous driving downstream tasks yielded promising results, the sample size remains limited, and a larger dataset is needed to draw definitive conclusions.

Future work. Future research endeavors will primarily focus on addressing the aforementioned limitations and enhancing the realism of rendered scenes, thereby further reducing the domain gap between rendered and recorded data. Specifically, we intend to explore hierarchical Gaussian Splatting to mitigate block-related artifacts. Utilizing the generated normal information, we will investigate methods to improve LiDAR intensity simulation, better reflecting the directional dependencies observed in real LiDAR data.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023. 4

[2] 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, 2020. 1

[3] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019. 1

[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. 2

[5] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, et al. Omnire: Omni urban scene reconstruction. *arXiv preprint arXiv:2408.16760*, 2024. 2, 7

[6] 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. 1

[7] Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. *arXiv preprint arXiv:2411.16816*, 2024. 2

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

[9] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2989–2998, 2023. 3

[10] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5): 922–923, 1976. 3

[11] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 3

[12] 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. 2, 4, 5

[13] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. Deviant: Depth equivariant network for monocular 3d object detection. In *European Conference on Computer Vision*, pages 664–683. Springer, 2022. 7

[14] Sivabalan Manivasagam, Ioan Andrei Bârsan, Jingkang Wang, Ze Yang, and Raquel Urtasun. Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8272–8282, 2023. 8

[15] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7210–7219, 2021. 4

[16] Tamás Matuszka, Iván Barton, Ádám Butykai, Péter Hajas, Dávid Kiss, Domonkos Kovács, Sándor Kunsági-Máté, Péter Lengyel, Gábor Németh, Levente Pető, et al. aimotive dataset: A multimodal dataset for robust autonomous driving with long-range perception. In *ICLR 2023 Workshop on SR4AD*, 2023. 1

[17] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[18] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–19, 2024. 5, 6

[19] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024. 4

[20] 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

[21] Chensheng Peng, Chengwei Zhang, Yixiao Wang, Chenfeng Xu, Yichen Xie, Wenzhao Zheng, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. Desire-gs: 4d street gaussians for static-dynamic decomposition and surface reconstruction for urban driving scenes. *arXiv preprint arXiv:2411.11921*, 2024. 2, 7

[22] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022. 4

[23] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[24] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1

[25] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022. 3

[26] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David

McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 4

[27] 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. 2

[28] 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. 3

[29] 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. 3

[30] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. *ArXiv*, abs/2403.17822, 2024. 4

[31] Yuehao Wang, Chaoyi Wang, Bingchen Gong, and Tianfan Xue. Bilateral guided radiance field processing. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 4

[32] 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

[33] 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. In *ECCV*, 2024. 2

[34] 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

[35] 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. *arXiv preprint arXiv:2409.06765*, 2024. 4

[36] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details in 3d gaussian splatting. In *ACM Multimedia 2024*, 2024. 4

[37] 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 autonomous driving, 2024. 7

[38] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 4

[39] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 3

[40] 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 (CVPR)*, pages 21634–21643, 2024. 2

[41] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 5