MARS: An Instance-aware, Modular and Realistic Simulator for Autonomous Driving

Zirui Wu[1,2], Tianyu Liu[1,3], Liyi Luo[1,4], Zhide Zhong[1,5], Jianteng Chen[1,5], Hongmin Xiao[1,6], Chao Hou[1,7], Haozhe Lou[1,8], Yuantao Chen[1,9], Runyi Yang[1,10], Yuxin Huang[1,5], Xiaoyu Ye[1,5], Zike Yan[1], Yongliang Shi[1], Yiyi Liao[11], and Hao Zhao[1]⋆

[1] AIR, Tsinghua University; [2] System Thurst, HKUST(GZ); [3] HKUST; [4] McGill University; [5] Beijing Institute of Technology; [6] National University of Singapore; [7] HKU; [8] University of Wisconsin Madison; [9] Xi'an University of Architecture and Technology; [10] Imperial College London; [11] Zhejiang University

Abstract. Nowadays, autonomous cars can drive smoothly in ordinary cases, and it is widely recognized that realistic sensor simulation will play a critical role in solving remaining corner cases by simulating them. To this end, we propose an autonomous driving simulator based upon neural radiance fields (NeRFs). Compared with existing works, ours has three notable features: (1) **Instance-aware**. Our simulator models the foreground instances and background environments separately with independent networks so that the static (e.g., size and appearance) and dynamic (e.g., trajectory) properties of instances can be controlled separately. (2) **Modular**. Our simulator allows flexible switching between different modern NeRF-related backbones, sampling strategies, input modalities, etc. We expect this modular design to boost academic progress and industrial deployment of NeRF-based autonomous driving simulation. (3) **Realistic**. Our simulator set new state-of-the-art photorealism results given the best module selection. Our simulator will be **open-sourced** while most of our counterparts are not. Project page: https://open-air-sun.github.io/mars/.

Keywords: Autonomous Driving Simulator · Neural Radiance Fields.

1 Introduction

Autonomous driving [10,11,28,14,22,12] is arguably the most important application of modern 3D scene understanding [4,23] techniques. Nowadays, Robotaxis can run in big cities with up-to-date HD maps, handling everyday driving scenarios smoothly. However, once a corner case that lies out of the distribution of an autonomous driving algorithm happens on the road unexpectedly, the lives of passengers are put at risk. The dilemma is that while we need more training data about corner cases, collecting them in the real world usually means danger and high expenses. To this end, the community believes that photorealistic

⋆ Corresponding author. `zhaohao@air.tsinghua.edu.cn`.
Sponsored by Tsinghua-Toyota Joint Research Fund (20223930097).

simulation [15,5,25,9] is a technical path of great potential. If an algorithm can experience enormous corner cases in a simulator with a small sim-to-real gap, the performance bottleneck of current autonomous driving algorithms can be potentially addressed.

Existing autonomous driving simulation methods have their own limitations. CARLA [7] is a widely used sensor simulator based upon traditional graphics engines, whose realism is restricted by asset modeling and rendering qualities. AADS [15] also exploits traditional graphics engines but demonstrates impressive photorealism using well-curated assets. On the other hand, GeoSim [5] introduces a data-driven scheme for realistic simulation by learning an image enhancement network. Flexible asset generation and rendering can be achieved through image composition with promisingly good geometry and realistic appearance.

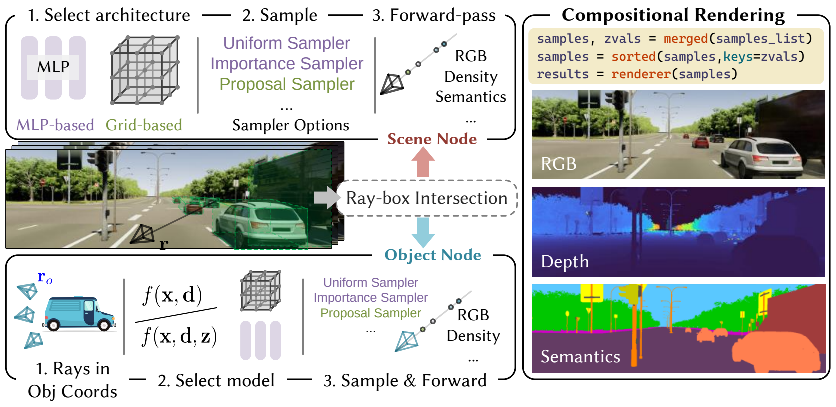
In this paper, we take advantage of the realistic rendering ability of NeRFs for autonomous driving simulation. Training data captured from real-world environments guarantees a small sim-to-real gap. Several works also exploit NeRFs to model cars [18] and static backgrounds [9] in outdoor environments. However, the inability to model complex dynamic scenes that are composed of both moving objects and static environments limits their practical use for real-world sensor simulation. Recently, Neural Scene Graph (NSG) [19] decomposes dynamic scenes into learned scene graphs and learns latent representations for category-level objects. However, its multi-plane-based representation for background modeling cannot synthesize images under large viewpoint changes.

To be specific, our central contribution is the very first **open-source** NeRF-based modular framework for photorealistic autonomous driving simulation. The proposed pipeline models foreground instances and background environments in a decomposed fashion. Different NeRF backbone architectures and sampling methods are incorporated in a unified manner with multi-modal inputs supported. The best module combination of the proposed framework achieves state-of-the-art rendering performance on public benchmarks with large margins, indicating photorealistic simulation results.

2 Method

Overview. As illustrated in Fig. 1, we aim to provide a modular framework for constructing compositional neural radiance fields, where realistic sensor simulation can be conducted for outdoor driving scenes. A large unbounded outdoor environment with plenty of dynamic objects is taken into consideration.

The input to the system consists of a set of RGB-images $\{\mathcal{I}_i\}^N$ (captured by vehicle-side or roadside sensors), sensor poses $\{\mathcal{T}_i\}^N$ (calculated using IMU/GPS signals), and object tracklets (including 3D bounding boxes $\{\mathcal{B}_{ij}\}^{N\times M}$, categories $\{\texttt{type}_{ij}\}^{N\times M}$, and instance IDs $\{\texttt{idx}_{ij}\}^{N\times M}$). N is the number of input frames and M is the number of tracked instances $\{\mathcal{O}_j\}^M$ across the whole sequence. An optional set of depth maps $\{\mathcal{D}_i\}^N$ and semantic segmentation masks $\{\mathcal{S}_i\}^N$ can also be adopted as extra supervision signals during training. By constructing a compositional neural field, the proposed framework can simulate



```
samples, zvals = merged(samples_list)
samples = sorted(samples,keys=zvals)
results = renderer(samples)
```

Fig. 1. Pipeline. **Left**: We first calculate the ray-box intersection of the queried ray $\mathbf{r}$ and all visible instance bounding boxes $\{\mathcal{B}_{ij}\}$. For the background node, we directly use the selected scene representation model and the chosen sampler to infer point-wise properties, as in conventional NeRFs. For the foreground nodes, the ray is first transformed into the instance frame as $\mathbf{r}_o$ before being processed through foreground node representations (Sec. 2.1). **Right**: All the samples are composed and rendered into RGB images, depth maps, and semantics (Sec. 2.2).

realistic sensor perception signals (including RGB images, depth maps, semantic segmentation masks, etc.) at given sensor poses. Instance editing on object trajectories and appearances is also supported.

Pipeline. Our framework model each foreground instance and the background node compositionally. As shown in Fig. 1, when querying properties (RGB, depth, semantics, etc.) of a given ray $\mathbf{r}$, we first calculate its intersection with all visible objects' 3D bounding boxes to get the entering and leaving distances $[t_{\texttt{in}}, t_{\texttt{out}}]$. Afterward, both the background node (Fig. 1 left-top) and the foreground object nodes (Fig. 1 left-bottom) are queried, where each node samples a set of 3D points and uses its specific neural representation network to obtain point properties (RGB, density, semantics, etc.). Specifically, to query foreground nodes, we convert the ray origins and directions from world space into instance frames according to the object tracklets. Finally, all the ray samples from background and foreground nodes are composed and volume-rendered to produce pixel-wise rendering results (Fig. 1 right, Sec. 2.2).

We observe that the nature of background nodes (typically unbounded large-scale scenes) differs from the object-centric foreground nodes, while current works [13,19] in sensor simulation use unified NeRF models. Our framework provides a flexible and open-sourced framework that supports different design choices of scene representations for background and foreground nodes and can easily incorporate new state-of-the-art methods of static scene reconstruction and object-centric reconstructions.

2.1 Scene Representation

We decompose the scene into a large-scale unbounded NeRF (as the background node) and multiple object-centric NeRFs (as independent foreground nodes). Conventionally, a neural radiance field maps a given 3D point coordinate $\mathbf{x} = (x, y, z), \mathbf{x} \in \mathbb{R}^3$ and a 2D viewing direction $\mathbf{d} \in \mathbb{S}^2$ to its radiance $\mathbf{c}$ and volume density σ shown in Eq. 1. Based upon this seminal representation, many variants have been proposed for different purposes, so we take a modular design.

$$f(\mathbf{x}, \mathbf{d}) = (\mathbf{c}, \sigma) : [\mathbb{R}^3, \mathbb{S}^2] \rightarrow [\mathbb{R}^3, \mathbb{R}^+] \tag{1}$$

The challenge of modeling unbounded background scene photo-realistically lies in accurately representing far regions, so we utilize the unbounded scene warping [2] to contract the far region. For foreground nodes, we support both the code-conditioned representation $f(\mathbf{x}, \mathbf{d}, \mathbf{z}) = (\mathbf{c}, \sigma)$ ($\mathbf{z} \in \mathbb{R}^k$ denotes the instance-wise latent code) and the conventional ones, which will be explained as follows.

Architectures. In our modular framework, we support various NeRF backbones, which can be roughly categorized into two hyper-classes: MLP-based methods [16,1,2], or grid-based methods that store spatially-variant features in their hash grid voxel vertices [17,21]. Although these architectures differ from each other in details, they follow the same high-level formulation of Eq. 1 and are capsuled in modules under a unified interface in MARS.

While the MLP-based representations are simple in mathematical form, we give a formal exposition of grid-based methods. The specific implementation of a multi-resolution feature grid $\{\mathcal{G}_\theta^l\}_{l=1}^L$ has layer-wise resolutions $R_l := \lfloor R_{\min} \cdot b^l \rfloor, b = \exp\left(\frac{\ln R_{\max} - \ln R_{\min}}{L-1}\right)$, where $R_{\min}, R_{\max}$ are the coarsest and the finest resolution [26,17]. The coordinates $\mathbf{x}$ are first scaled to each resolution before being processed by the ceiling and flooring operations to $\lceil \mathbf{x} \cdot R_l \rceil, \lfloor \mathbf{x} \cdot R_l \rfloor$ and hashed to obtain table indexes [17]. The extracted feature vectors are then tri-linearly interpolated and decoded through a shallow MLP.

$$(\mathbf{c}, \sigma) = f_\theta\left(\texttt{interp}(\texttt{hash\_and\_lookup}(\mathbf{x}, \{\mathcal{G}_\theta^l\}_{l=1}^L)), \mathbf{d}\right). \tag{2}$$

Sampling. We support various sampling strategies, including the recently proposed proposal network [2], which distills a density field from a radiance-free NeRF model to generate ray samples and other sampling schemes like coarse-to-fine sampling [16] or uniform sampling [8] for flexibility.

Foreground Nodes. For rendering foreground instances, we first transform the projected rays into per-instance coordinate space and then infer the object-centric NeRFs in each instance-wise canonical space. The default setting of our framework uses code-conditioned models that exploit latent codes to encode instance features and **shared** category-level decoders to encode class-wise priors, allowing the modeling of many long tracklets with compact memory usage. Meanwhile, the conventional ones without code conditions are also supported in our framework. We detailed our modified foreground representation (denoted as 'Ours' in Sec. 3) in supplementary materials.

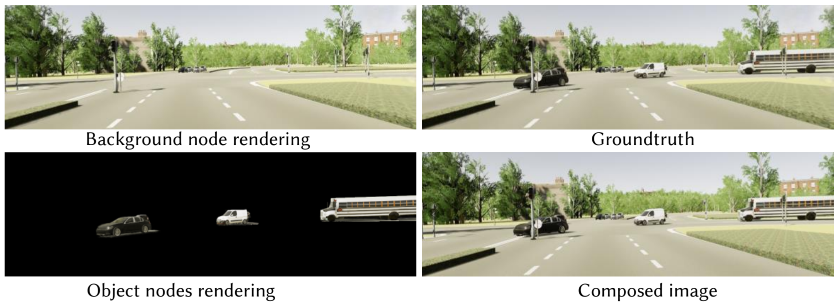


Fig. 2. Illustration on the compositional rendering. Some of the static vehicles in the far region are considered as background objects.

2.2 Compositional Rendering

Figure 2 demonstrates the compositional rendering results. To render an image at a given camera pose $\mathcal{T}_i$, we cast a ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ at each rendered pixel. For each ray $\mathbf{r}$, we first calculate the intersection interval $[t_{\text{in}}, t_{\text{out}}]$ with all visible foreground nodes $\mathcal{O}_{ij}$ (Fig. 3) and transform the samples $\{P_k^{\text{obj-j}}\}$ along the ray from world space into each foreground canonical space. We also sample a set of 3D points along the ray ($\{P_k^{\text{bg}}\}$ as background samples. Samples in all nodes are first passed through their corresponding networks to obtain point-wise colors $\{\mathbf{c}_k^{\text{bg, obj}}\}$, densities $\{\sigma_k^{\text{bg, obj}}\}$, and foreground semantic logits $\{\mathbf{s}_k^{\text{bg}}\}$. Considering that the semantic properties of foreground samples are actually their category label, we create a one-hot vector as:

$$\mathbf{s}_k^{\text{obj-j}}[l] = \begin{cases} \sigma_k^{\text{obj-j}} & \text{if } l = \text{category of j's instance} \\ 0 & \text{otherwise} \end{cases}, \text{for } l \text{ in category}. \tag{3}$$

To aggregate the point-wise properties, we sort all the samples by their ray distance in world space and use the standard volume rendering process to render pixel-wise properties:

$$\hat{\mathbf{c}}(\mathbf{r}) = \sum_{P_i} T_i \alpha_i \mathbf{c}_i + (1 - \texttt{accum}) \cdot \mathbf{c}_{\text{sky}}, \quad T_i = \exp(-\sum_{k=1}^{i-1} \sigma_k \delta_k), \tag{4}$$

$$\hat{d}(\mathbf{r}) = \sum_{P_i} T_i \alpha_i t_i + (1 - \texttt{accum}) \cdot \texttt{inf}, \; \hat{\mathbf{s}}(\mathbf{r}) = \sum_{P_i} T_i \alpha_i \mathbf{s}_i + (1 - \texttt{accum}) \cdot \mathbf{s}_{\text{sky}}, \tag{5}$$

where $P_i \in \texttt{sorted}(\{P_i^{\text{bg, obj}}\}), \alpha_i = 1 - \exp(-\sigma_i \delta_i), \delta_i = t_{i+1} - t_i$, $\texttt{accum} = \sum_{P_i} T_i \alpha_i$, $\mathbf{c}_{\text{sky}}$ is the rendered color from the Sky model (Sec.2.3), `inf` is the upper bound distance, and $\mathbf{s}_{\text{sky}}$ is the one-hot semantic logits of the `sky` category.

2.3 Towards Realistic Rendering

Sky Modeling. In our framework, we support the usage of a sky model to deal with appearances at infinite distance, where an MLP-based spherical environment map [20] is leveraged to model the infinitely far regions that never intersect

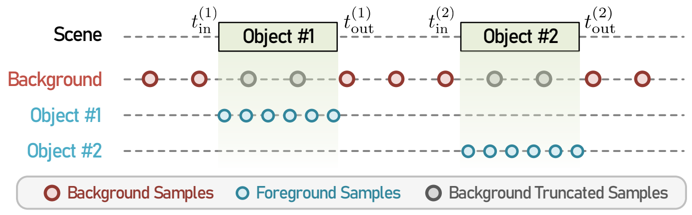


Fig. 3. Visual demonstration on our **conflict-free** sampling process. We use uniform sampling in all nodes for illustration.

opaque surfaces:

$$f_{\text{sky}}(\mathbf{d}) = \mathbf{c}_{\text{sky}} : \mathbb{S}^2 \to \mathbb{R}^3 \tag{6}$$

However, naïvely blending the sky color $\mathbf{c}_{\text{sky}}$ with background and foreground rendering (Eq. 4) leads to potential inconsistency. Therefore, we introduce a BCE semantic regularization to alleviate this issue:

$$\mathcal{L}_{\text{sky}} = \texttt{BCE}(1 - \texttt{accum}, \mathcal{S}_{\text{sky}}). \tag{7}$$

Resolving Conflict Samples. Due to the fact that our background and foreground sampling are done independently, there is a chance that background samples fall within the foreground bounding box (Fig. 3 Background Truncated Samples). The compositional rendering may mistakenly classify foreground samples as background (referred to later as background-foreground ambiguity). As a result, after removing the foreground instance, artifacts will emerge in the background area (Fig. 4). Ideally, with sufficient multi-view supervision signal, the system can automatically learn to distinguish between foreground and background during the training process. However, for a data-driven simulator, obtaining abundant and high-quality multi-view images is challenging for users as vehicles move fast on the road. The ambiguity is **NOT** observed in NSG [19] as NSG only samples a few points on the ray-plane intersections, and is unlikely to have much background truncated samples.

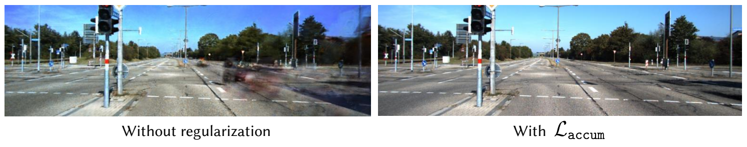


Fig. 4. We show that the background truncated samples cause background-foreground ambiguity without our regularization.

To address this issue, we devise a regularization term that minimizes the density sum of background truncated samples to minimize their influence during the rendering process as:

$$\mathcal{L}_{\texttt{accum}} = \sum_{P_i^{(\text{tr})}} \sigma_i, \tag{8}$$

where $\{P_i^{(\text{tr})}\}$ denotes background truncated samples.

2.4 Optimization

To optimize our system, we minimize the following objective function:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{color}} + \lambda_2 \mathcal{L}_{\text{depth}} + \lambda_3 \mathcal{L}_{\text{sem}} + \lambda_4 \mathcal{L}_{\text{sky}} + \lambda_5 \mathcal{L}_{\texttt{accum}}, \tag{9}$$

where λ_{1-5} are weighting parameters. $\mathcal{L}_{\text{sky}}$ and $\mathcal{L}_{\texttt{accum}}$ are explained in Eq. 7 and 8.

Color Loss: we adopt a standard MSE loss that minimizes the photo-metric errors as:

$$\mathcal{L}_{\text{color}} = ||\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}(\mathbf{r})||_2^2. \tag{10}$$

Depth Loss: We introduce a depth loss to address textureless regions and regions that are observed from sparse viewpoints. We have devised two strategies for supervising the geometry. Given depth data, we utilize a ray distribution loss derived from [6]. On the other hand, if the depth data is not available, we utilize a mono-depth network and apply mono-depth loss following [26].

$$\mathcal{L}_{\text{depth}} = \begin{cases} \mathcal{L}_{\text{sensor\_depth}} & \text{if depth data is available} \\ \mathcal{L}_{\text{mono\_depth}} & \text{if depth data is not available} \end{cases} \tag{11}$$

Semantic Losses: we follow SemanticNeRF [29] and use a cross-entropy semantic loss $\mathcal{L}_{\text{sem}} = \texttt{CrossEntropy}(\mathbf{s}(\mathbf{r}), \mathcal{S}(\mathbf{r}))$.

3 Experiments

In this section, we provide extensive experimental results to demonstrate the proposed instance-aware, modular, and realistic simulator for autonomous driving. We evaluate our method on scenes from the KITTI [10] dataset and the Virtual KITTI-2 (V-KITTI) [3] dataset. In the following, we use **"our default setting"** to denote a grid-based NeRF with proposal sampler for the background node, and our modified category-level representation with coarse-to-fine sampler for foreground nodes.

Table 1. Qunatitative results on image reconstruction task & Comparisons on the settings with baseline methods. The dataset used for evaluation is KITTI.

	NeRF [16]	NeRF+Time	NSG [19]	PNF [13]	SUDS [24]	Ours
PSNR ↑	23.34	24.18	26.66	27.48	28.31	**29.06**
SSIM ↑	0.662	0.677	0.806	0.870	0.876	**0.885**
Instance-aware	×	×	✓	✓	×	✓
Modular	×	×	×	×	×	✓
Open-sourced	✓	-	✓	×	✓	✓

3.1 Photorealistic Rendering

We validate the photorealistic rendering performance of our simulator by evaluating image reconstruction and novel view synthesis (NVS) following [19,24].

Table 2. Qunatitative results on novel view synthesis

	KITTI-75%			**KITTI-50%**			**KITTI-25%**		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [16]	18.56	0.557	0.554	19.12	0.587	0.497	18.61	0.570	0.510
NeRF+Time	21.01	0.612	0.492	21.34	0.635	0.448	19.55	0.586	0.505
NSG [19]	21.53	0.673	0.254	21.26	0.659	0.266	20.00	0.632	0.281
SUDS [24]	22.77	0.797	0.171	23.12	**0.821**	**0.135**	20.76	0.747	0.198
Ours	**24.23**	**0.845**	**0.160**	**24.00**	0.801	0.164	**23.23**	**0.756**	**0.177**
	+1.46	+0.048	-0.011	+0.88	-0.020	+0.029	+2.47	+0.009	-0.021
	VKITTI-75%			**VKITTI-50%**			**VKITTI-25%**		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [16]	18.67	0.548	0.634	18.58	0.544	0.635	18.17	0.537	0.644
NeRF+Time	19.03	0.574	0.587	18.90	0.565	0.610	18.04	0.545	0.626
NSG [19]	23.41	0.689	0.317	23.23	0.679	0.325	21.29	0.666	0.317
SUDS [24]	23.87	0.846	0.150	23.78	0.851	0.142	22.18	0.829	0.160
Ours	**29.79**	**0.917**	**0.088**	**29.63**	**0.916**	**0.087**	**27.01**	**0.887**	**0.104**
	+5.92	+0.071	-0.062	+5.85	+0.065	-0.055	+4.83	+0.058	-0.056

Baselines. We conduct qualitative and quantitative comparisons against other state-of-the-art methods: NeRF [16], NeRF with timestamp input (denoted as NeRF+Time), NSG [19], PNF [13], and SUDS [24]. Note that none of them simultaneously meet all three standards mentioned in Table. 1.

Implementation Details. Our model is trained for 200,000 iterations with 4096 rays per batch, using RAdam as optimizers. The learning rate of the background node is assigned $1 * 10^{-3}$ decaying to $1 * 10^{-5}$, while that of $5 * 10^{-3}$ decaying to $1 * 10^{-5}$ in object nodes.

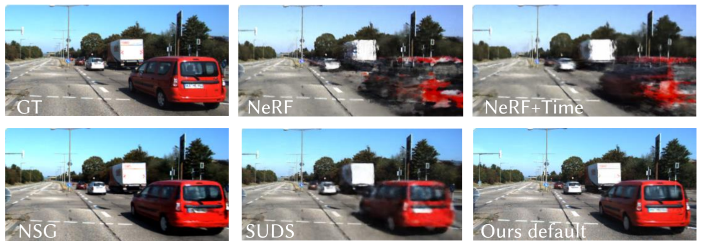


Fig. 5. Qualitative image reconstruction results on KITTI dataset.

Experiment Settings. The training and testing image sets in the image reconstruction setting are identical, while in the NVS task, we render the frames that are not included in the training data. Specifically, we hold out every 4th frames, every 2nd and 4th frames, and training with only one in every four frames, namely 25%, 50%, and 75%.

We follow the standard evaluation protocol in image synthesis and report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [27] of our default setting for quantitative evaluations. Results are shown in Table 1 for image reconstruction and Table 2 for NVS, which indicate that our method outperforms baseline methods in both settings. We can achieve 29.79 PSNR on V-KITTI using 75% training data, while the best result previously published is 23.87.

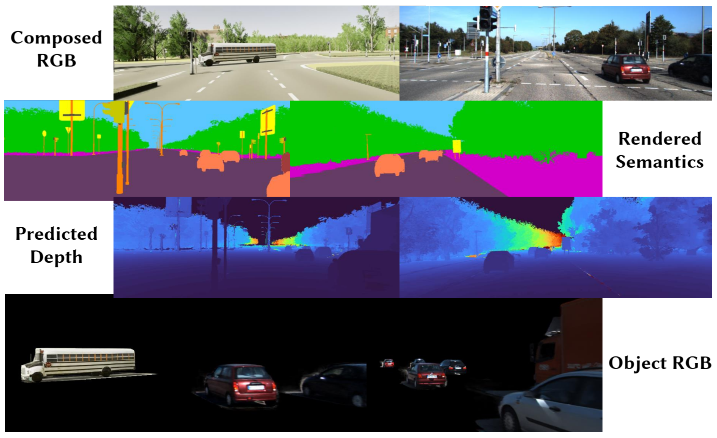


Fig. 6. Gallery of different rendering channels.

3.2 Instance-wise Editing

Our framework separately models background and foreground nodes, which allows us to edit the scene in an instance-aware manner. We qualitatively present our capability to remove instances, add new instances, and edit vehicle trajectories. In Fig. 7, we show some editing examples of rotating and translating a vehicle, though more results can be found in our video clip.

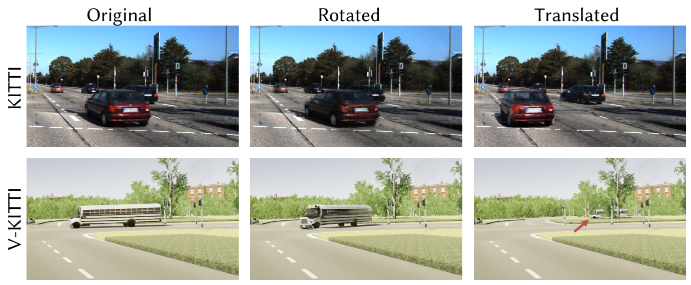


Fig. 7. Rendering results on the edited scene.

3.3 The blessing of moduler design

We use different combinations of background and foreground nodes, samplers, and supervision signals for evaluation, which is credited to our modular design.

Note that some of the baseline methods in the literature actually correspond to an ablation entry in this table. For instance, PNF [13] uses NeRF as background node representation and instance-wise NeRF as foreground node representation with semantic losses. NSG [19] uses NeRF as background node representation and category-level NeRF as foreground representation, but with a multi-plane sampling strategy. Our default setting uses grid-based background node representation, and our proposed category-level method for foreground node representation.

3.4 Ablation Results

In this section, we analyze different experiment settings, verifying the necessity of our design. We reveal the impact of different design choices in background node representation, foreground node representation, etc. Specifically, we present all experiments with 50,000 iterations. Unlike prior works [24,13,19] that evaluate their method on a short sequence of 90 images, we use the full sequence from the dataset for all evaluation. Since they are not open-sourced and their exact evaluation sequences are not known, we hope our new benchmarking would standardize this important field. Quantitative evaluation can be found in Table 3.

For background and foreground nodes, we substitute our default model (ID 1 in Table 3) with MLP-based and grid-based model and list their metrics in row 2, 7-12. In the 3rd-6th row, we show the effectiveness of our model components. For model and sampler, selected modules for background and foreground nodes are noted before and after the slash, respectively.

Table 3. Quantitative evaluation for ablation studies

ID	Settings							KITTI			V-KITTI		
	Model	Sampler	Category	$\mathcal{L}_{\text{sky}}$	$\mathcal{L}_{\text{depth}}$	$\mathcal{L}_{\text{sem}}$	$\mathcal{L}_{\text{accum}}$	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
1*	Grid / Ours	prop / c2f †						**25.04**	**0.782**	**0.175**	**28.37**	**0.907**	**0.108**
2	**MLP** / Ours	**c2f** / c2f						20.14	0.589	0.476	22.19	0.664	0.409
3	Grid / Ours	prop / c2f					×	21.35	0.713	0.242	27.30	0.881	0.130
4	Grid / Ours	prop / c2f		×		×		23.68	0.774	0.181	27.32	0.881	0.129
5	Grid / Ours	prop / c2f			×			23.66	0.769	0.184	27.30	0.880	0.128
6	Grid / Ours	prop / c2f		×				20.07	0.723	0.251	27.42	0.863	0.148
7	Grid / **MLP**	prop / **c2f**						20.46	0.709	0.255	26.46	0.875	0.132
8	Grid / **Grid**	prop / **prop**						22.23	0.741	0.211	25.22	0.871	0.134
9	Grid / **MLP**	prop / **c2f**	×					20.98	0.699	0.257	27.27	0.881	0.130
10	Grid / **Grid**	prop / **prop**	×					23.71	0.763	0.193	26.65	0.882	0.125
11*	**MLP** / **MLP**	**c2f** / **c2f**						20.42	0.592	0.472	21.77	0.659	0.410

† prop stands for proposal sampler, and c2f stands for coarse-to-fine sampler.
* ID 1 is our default setting. ID 11 is similar to the setting of NSG [19] with coarse-to-fine sampler instead.

4 Conclusion

In this paper, we present a modular framework for photorealistic autonomous driving simulation based on NeRFs. Our **open-sourced** framework consists of a background node and multiple foreground nodes, enabling the modeling of complex dynamic scenes. We demonstrate the effectiveness of our framework through extensive experiments. The proposed pipeline achieved state-of-the-art rendering performance on public benchmarks. We also support different design choices of scene representations and sampling strategies, offering flexibility and versatility in the simulation process.

Limitations. Our method requires hours to train and is not capable of rendering in real-time. Besides, our method fails to consider the dynamic specular effects on glasses or other reflective materials that may cause artifacts in rendered images. Improving simulation efficiency and view-dependent effects will be our future work.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5835–5844 (Oct 2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. arXiv (Mar 2022)
3. Cabon, Y., Murray, N., Humenberger, M.: Virtual KITTI 2, http://arxiv.org/abs/2001.10773
4. Chen, X., Zhao, H., Zhou, G., Zhang, Y.Q.: PQ-Transformer: Jointly Parsing 3D Objects and Layouts From Point Clouds. IEEE Robotics and Automation Letters **7**(2), 2519–2526 (Apr 2022)
5. Chen, Y., Rong, F., Duggal, S., Wang, S., Yan, X., Manivasagam, S., Xue, S., Yumer, E., Urtasun, R.: GeoSim: Realistic Video Simulation via Geometry-Aware Composition for Self-Driving, http://arxiv.org/abs/2101.06543
6. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised NeRF: Fewer Views and Faster Training for Free. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12872–12881 (Jun 2022)
7. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An Open Urban Driving Simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16. PMLR (Oct 2017)
8. Fridovich-Keil, S., Meanti, G., Warburg, F., Recht, B., Kanazawa, A.: K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In: Computer Vision And Pattern Recognition, 2023 (2023)
9. Fu, X., Zhang, S., Chen, T., Lu, Y., Zhu, L., Zhou, X., Geiger, A., Liao, Y.: Panoptic NeRF: 3D-to-2D Label Transfer for Panoptic Urban Scene Segmentation. In: 2022 International Conference on 3D Vision (3DV). pp. 1–11 (Sep 2022)
10. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research **32**(11), 1231–1237 (2013)
11. Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., Lu, L., Jia, X., Liu, Q., Dai, J., Qiao, Y., Li, H.: Planning-Oriented Autonomous Driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17853–17862 (2023)
12. Jin, B., Liu, X., Zheng, Y., Li, P., Zhao, H., Zhang, T., Zheng, Y., Zhou, G., Liu, J.: ADAPT: Action-aware Driving Caption Transformer. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 7554–7561 (May 2023)
13. Kundu, A., Genova, K., Yin, X., Fathi, A., Pantofaru, C., Guibas, L.J., Tagliasacchi, A., Dellaert, F., Funkhouser, T.: Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12871–12881 (2022)
14. Li, P., Zhao, R., Shi, Y., Zhao, H., Yuan, J., Zhou, G., Zhang, Y.Q.: LODE: Locally Conditioned Eikonal Implicit Scene Completion from Sparse LiDAR. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). arXiv (Feb 2023)
15. Li, W., Pan, C.W., Zhang, R., Ren, J.P., Ma, Y.X., Fang, J., Yan, F.L., Geng, Q.C., Huang, X.Y., Gong, H.J., Xu, W.W., Wang, G.P., Manocha, D., Yang, R.G.: AADS: Augmented autonomous driving simulation using data-driven algorithms. Science Robotics **4**(28), eaaw0863 (Mar 2019)

16. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 405–421. Lecture Notes in Computer Science, Springer International Publishing, Cham (2020)
17. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics **41**(4), 1–15 (Jul 2022)
18. Niemeyer, M., Geiger, A.: GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11453–11464 (2021)
19. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural Scene Graphs for Dynamic Scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. arXiv (Mar 2021)
20. Rematas, K., Liu, A., Srinivasan, P., Barron, J., Tagliasacchi, A., Funkhouser, T., Ferrari, V.: Urban Radiance Fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12922–12932 (Jun 2022)
21. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A Modular Framework for Neural Radiance Field Development. ACM Transactions on Graphics **1**(1) (May 2023)
22. Tian, B., Liu, M., Gao, H.a., Li, P., Zhao, H., Zhou, G.: Unsupervised road anomaly detection with language anchors. In: 2023 IEEE international conference on robotics and automation (ICRA). pp. 7778–7785 (2023)
23. Tian, B., Luo, L., Zhao, H., Zhou, G.: VIBUS: Data-efficient 3D Scene Parsing with VIewpoint Bottleneck and Uncertainty-Spectrum Modeling. Journal of Photogrammetry and Remote Sensing (Oct 2022)
24. Turki, H., Zhang, J.Y., Ferroni, F., Ramanan, D.: SUDS: Scalable Urban Dynamic Scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. arXiv (Mar 2023)
25. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: UniSim: A Neural Closed-Loop Sensor Simulator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1389–1399 (2023)
26. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. In: Advances in Neural Information Processing Systems (Oct 2022)
27. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (Jun 2018)
28. Zheng, Y., Zhong, C., Li, P., Gao, H.a., Zheng, Y., Jin, B., Wang, L., Zhao, H., Zhou, G., Zhang, Q., Zhao, D.: STEPS: Joint Self-supervised Nighttime Image Enhancement and Depth Estimation. In: 2023 IEEE Conference on Robotics and Automation (ICRA 2023) (Feb 2023)
29. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-Place Scene Labelling and Understanding with Implicit Scene Representation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (Aug 2021)