Street Gaussians: Modeling Dynamic Urban Scenes with Gaussian Splatting

Yunzhi Yan[1,2], Haotong Lin[1], Chenxu Zhou[1], Weijie Wang[1], Haiyang Sun[2], Kun Zhan[2], Xianpeng Lang[2], Xiaowei Zhou[1], and Sida Peng[1(✉)]

[1] Zhejiang University, Hangzhou, China
[2] Li Auto, Beijing, China

Abstract. This paper aims to tackle the problem of modeling dynamic urban streets for autonomous driving scenes. Recent methods extend NeRF by incorporating tracked vehicle poses to animate vehicles, enabling photo-realistic view synthesis of dynamic urban street scenes. However, significant limitations are their slow training and rendering speed. We introduce Street Gaussians, a new explicit scene representation that tackles these limitations. Specifically, the dynamic urban scene is represented as a set of point clouds equipped with semantic logits and 3D Gaussians, each associated with either a foreground vehicle or the background. To model the dynamics of foreground object vehicles, each object point cloud is optimized with optimizable tracked poses, along with a 4D spherical harmonics model for the dynamic appearance. The explicit representation allows easy composition of object vehicles and background, which in turn allows for scene editing operations and rendering at 135 FPS (1066 * 1600 resolution) within half an hour of training. The proposed method is evaluated on multiple challenging benchmarks, including KITTI and Waymo Open datasets. Experiments show that the proposed method consistently outperforms state-of-the-art methods across all datasets. The code will be released to ensure reproducibility.

Keywords: 3D Gaussians · View Synthesis · Real-Time Rendering

1 Introduction

Modeling dynamic 3D streets from images has many important applications, such as city simulation, autonomous driving, and gaming. For instance, the digital twin of city streets can be used as the simulation environment for self-driving vehicles, thereby reducing the training and test costs. These applications require us to efficiently reconstruct 3D street models from captured data and render high-quality novel views in real-time (Fig. 1).

With the development of neural scene representations, there have been some methods [30,36,41,48,74] that attempt to reconstruct street scenes with neural

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-73464-9_10.

A. Leonardis et al. (Eds.): ECCV 2024, LNCS 15131, pp. 156–173, 2025.
https://doi.org/10.1007/978-3-031-73464-9_10

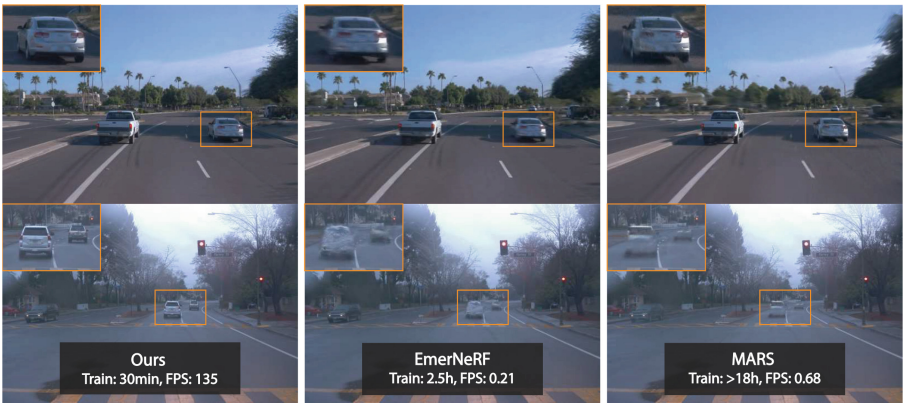

Fig. 1. Rendering results on the Waymo dataset [47]. Our method produces high-quality rendering at 135 FPS (1066×1600) within half an hour of training. Current SOTA methods [60,65] suffer from high training and rendering cost.

radiance fields [33]. To improve the modeling capability, Block-NeRF [48] divides the scene into several blocks and represents each one with a NeRF network. Although this strategy enables photo-realistic rendering of large-scale street scenes, Block-NeRF suffers from long training time due to the large amount of network parameters. Moreover, it cannot handle dynamic vehicles on the street, which are crucial aspects in autonomous driving environment simulation.

Recently, some methods [22,37,60,67] propose to represent dynamic driving scenes as compositional neural representations that consist of foreground moving cars and static background. To handle the dynamic car, they leverage tracked vehicle poses to establish the mapping between the observation space and the canonical space, where they use NeRF networks to model the car's geometry and appearance. Although these methods produce reasonable results, they are still limited to the high training cost and low rendering speed.

In this work, we propose a novel explicit scene representation for reconstructing dynamic 3D street scenes from images. The basic idea is utilizing point clouds to build dynamic scenes, which significantly increases the training and rendering efficiency. Specifically, we decompose urban street scenes into the static background and moving vehicles, which are separately built based on 3D Gaussians [19]. To handle the dynamics of foreground vehicles, we model their geometry as a set of points with optimizable tracked vehicle poses, where each point stores learnable 3D Gaussian parameters. Furthermore, the time-varying appearance is represented by a 4D spherical harmonics model that uses a time series function to predict spherical harmonics coefficients at any time step. Thanks to the dynamic 3D Gaussians representation, we can faithfully reconstruct the target urban street within half an hour and achieve real-time rendering (135FPS@1066×1600). Building upon the proposed scene representation, we develop several strategies to further improve the rendering performance,

including the tracked pose optimization,point cloud initialization, and sky modeling.

We evaluate the proposed method on Waymo Open [47] (Waymo) and KITTI [15] datasets, which present dynamic street scenes with complex vehicle motions and various environment conditions. Across all datasets, our approach achieves state-of-the-art performance in terms of rendering quality, while being rendered over 100 times faster than previous methods [37,60,65]. Furthermore, detailed ablations and scene editing applications are conducted to demonstrate the effectiveness of proposed components and the flexibility of the proposed representation, respectively.

Overall, this work makes the following contributions:

- We propose Street Gaussians, a novel scene representation for modeling complex dynamic street scenes, which efficiently reconstructs and renders high-fidelity urban street scenes in real-time.
- We propose several strategies including 4D spherical harmonics appearance model, tracked pose optimization, and point cloud initialization, which largely improve the rendering performance of Street Gaussians.

2 Related Work

Static Scene Modeling. Neural scene representation proposes to represent 3D scenes with neural networks, which can model complex scenes from images through differentiable rendering. NeRF [3–5,33,34] represents continuous volumetric scenes with MLP networks and achieves impressive rendering results. Some works have been proposed to extend NeRF to urban scenes [9,16,18,29,30,36,41,48,51]. GridNeRF [62] proposes muti-resolution feature planes to help NeRF generate photorealistic results on large-scale scenes. DNMP [30] models the scene with deformable mesh primitives initialized by voxelizing point clouds. NeuRas [29] takes scaffold mesh as input and optimizes the neural texture field to perform fast rasterization.

Point-based rendering works [1,10,21,26,42] define learned neural descriptors on point clouds and perform differentiable rasterization with a neural renderer. However, they require dense point clouds as input and generate blurry results under regions with low point counts. A very recent work 3D Gaussian Splatting (3D GS) [19] defines a set of anisotropic Gaussians in 3D world and performs adaptive density control to achieve high-quality rendering results with only sparse point clouds input. However, 3D GS assumes the scene to be static and can not model dynamic moving objects.

Dynamic Scene Modeling. Recent methods build 4D neural scene representation on single-object scenes by encoding time as additional input [2,13,25,27,28,38,39,46]. Some works learn a scene decomposition of outdoor scenes under the supervision of optical flow [52] or vision transformer feature [65]. However, their scene representation is not instance aware, limiting the applications for autonomous driving simulation. Another line of works model the scene as the composition of moving object models and a background model

[22,37,50,60,61,67] with neural fields, which is most similar to us. However, they suffer from high memory cost on large scale scene and can not perform real-time rendering.

Extending point-based rendering to dynamic scene is also investigated recently [64,72]. Recent approaches extend 3D GS to small-scale dynamic scenes by introducing deformation field [57,71], physical priors [31] or 4D parametrization [69] to 3D Gaussian model. More recently, some concurrent works [8,75] also explore 3D Gaussians in urban street scenes. DrivingGaussian [75] introduces Incremental 3D Static Gaussians and Composite Dynamic Gaussian Graphs. PVG [8] utilizes Periodic Vibration 3D Gaussians to model dynamic urban scene.

Simulation Environments for Autonomous Driving. Existing self-driving simulation engines such as CARLA [11] or AirSim [44] suffer from costly manual effort to create virtual environments and the lack of realism in the generated data. In recent years, a lot of effort has been put into building sensor simulations from autonomous driving data captured in real scenes. Some works [12,32,70] concentrate on LiDAR simulation by aggregating LiDAR and reconstructing textured primitives. However, they have difficulty handling high-resolution images and usually produce noisy appearance. Other works [7,53,68] reconstruct objects from multi-view images and LiDAR input, which can be interacted with other environments. However, these methods are restricted to existing images and fail to render novel views. Some methods utilize neural fields to perform multiply tasks including view synthesis [17,37,67], perception [14,22,74], generation [24,35,45,63,66] and inverse rendering [40,54–56] on driving scenes. However, they struggle with high training and rendering cost. In contrast, Our method focuses on performing real-time rendering of dynamic urban scenes, which is crucial for autonomous driving simulation.

3 Method

Given a sequence of images captured from a moving vehicle in an urban street scene, our goal is to develop a model capable of generating photorealistic images for view synthesis. Towards this objective, we propose a novel scene representation, named Street Gaussians, specifically designed for representing dynamic street scenes. As shown in the Fig. 2, we represent a dynamic urban street scene as a set of point clouds, each corresponding to either the static background or a moving vehicle (Sect. 3.1). The explicit point-based representation allows easy composition of separate models, enabling real-time rendering as well as the decomposition of foreground objects for editing applications (Sect. 3.2). The proposed scene representation can be effectively trained along with tracked vehicle poses from an off-the-shelf tracker, enhanced by our pose optimization strategy (Sect. 3.3).

3.1 Street Gaussians

In this section, we seek to find a dynamic scene representation that can be quickly constructed and rendered in real-time. Previous methods [22,60] typi-

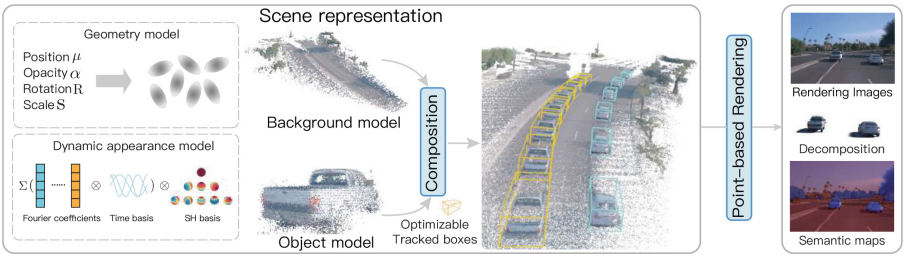

Fig. 2. Overview of Street Gaussians. The dynamic urban street scene is represented as a set of point-based background and foreground objects with optimizable tracked vehicle poses. Each point is assigned with a 3D Gaussian [19] including position, opacity, and covariance consisting of rotation and scale to represent the geometry. To represent the appearance, we assign each background point with a spherical harmonics model while the foreground points are associated with a dynamic spherical harmonics model. The explicit point-based representation allows easy composition of separate models, which enables real-time rendering of high-quality images and semantic maps (optional if 2D semantic information is provided during training), as well as the decomposition of foreground objects for editing applications.

cally face challenges with low training and rendering speed as well as accurate tracked vehicle poses. To tackle this problem, we propose a novel explicit scene representation, named Street Gaussians, which is built upon 3D Gaussians [19]. In Street Gaussians, we represent the static background and each moving vehicle object with a separate neural point cloud.

In the following, we will first focus on the background model, elaborating on several common attributes that are shared with the object model. Subsequently, we will delve into the dynamic aspects of the object model's design.

Background Model. The background model is represented as a set of points in the world coordinate system. Each point is assigned with a 3D Gaussian to softly represent the continuous scene geometry and color. The Gaussian parameters consist of a covariance matrix $\boldsymbol{\Sigma}_b$ and a position vector $\boldsymbol{\mu}_b \in \mathbb{R}^3$, which denotes the mean value. To avoid invalid value during optimization, each covariance matrix is further reduced to a scaling matrix $\mathbf{S}_b$ and a rotation matrix $\mathbf{R}_b$, where $\mathbf{S}_b$ is characterized by its diagonal elements and $\mathbf{R}_b$ is converted into a unit quaternion. The covariance matrix $\boldsymbol{\Sigma}_b$ can be recovered from $\mathbf{S}_b$ and $\mathbf{R}_b$ as:

$$\boldsymbol{\Sigma}_b = \mathbf{R}_b \mathbf{S}_b \mathbf{S}_b{}^T \mathbf{R}_b{}^T. \tag{1}$$

Apart from the position and covariance matrix, each Gaussian is also assigned with an opacity value $\alpha_b \in \mathbb{R}$ and a set of spherical harmonics coefficients $\mathbf{z}_b = (z_{m,l})_{l:0\le\ell\le\ell_{max}}^{m:-\ell\le m\le\ell}$ to represent scene geometry and appearance. To obtain the view-dependent color, the spherical harmonics coefficients are further multiplied by the spherical harmonics basis functions projected from the view direction. To represent 3D semantic information, each point is added with a semantic logit $\boldsymbol{\beta}_b \in \mathbb{R}^M$, where M is the number of semantic classes.

Object Model. Consider a scene containing N moving foreground object vehicles. Each object is represented with a set of optimizable tracked vehicle poses and a point cloud, where each point is assigned a 3D Gaussian, semantic logits, and a dynamic appearance model.

The Gaussian properties of both the object and the background are similar, sharing the same meaning for opacity α_o and scale matrix $\mathbf{S}_o$. However, their position, rotation, and appearance models differ from those of the background model. The position $\boldsymbol{\mu}_o$ and rotation $\mathbf{R}_o$ are defined in the object local coordinate system. To transform them into the world coordinate system (the background's coordinate system), we introduce the definition of tracked poses for objects. Specifically, the tracked poses of vehicles are defined as a set of rotation matrices $\{\mathbf{R}_t\}_{t=1}^{N_t}$ and translation vectors $\{\mathbf{T}_t\}_{t=1}^{N_t}$, where N_t represents the number of frames. The transformation can be defined as:

$$\begin{aligned} \boldsymbol{\mu}_w &= \mathbf{R}_t \boldsymbol{\mu}_o + \mathbf{T}_t, \\ \mathbf{R}_w &= \mathbf{R}_t \mathbf{R}_o, \end{aligned} \tag{2}$$

where $\boldsymbol{\mu}_w$ and $\mathbf{R}_w$ are the position and rotation of the corresponding object Gaussian in the world coordinate system, respectively. After transformation, the object's covariance matrix $\boldsymbol{\Sigma}_w$ can be obtained by Eq. 1 with $\mathbf{R}_w$ and $\mathbf{S}_o$. Note that we also found the tracked vehicle poses from the off-the-shelf tracker to be noisy. To address this issue, we treat the tracked vehicle poses as learnable parameters. We detail it in Sect. 3.3.

Simply representing object appearance with the spherical harmonics coefficients is insufficient for modeling the appearance of moving vehicles, as shown in Fig. 3, because the appearance of a moving vehicle is influenced by its position in the global scene. One straightforward solution is to use separate spherical harmonics to represent the object for each timestep. However, this representation will significantly increase the storage cost. Instead, we introduce the 4D spherical harmonics model by replacing each SH coefficient $z_{m,l}$ with a set of fourier transform coefficients $\boldsymbol{f} \in \mathbb{R}^k$ where k is the number of fourier coefficient. Given timestep t, $z_{m,l}$ is recovered by performing real-valued Inverse Discrete Fourier Transform:

$$z_{m,l} = \sum_{i=0}^{k-1} \boldsymbol{f}_i \cos\left(\frac{i\pi}{N_t} t\right). \tag{3}$$

With the proposed model, we encode time information into appearance without high storage cost.

The semantic representation of the object model is different from that of background. The main difference is that the semantic of the object model is a learnable one-dimensional scalar β_o which represents the vehicle semantic class from the trakcer instead of a M-dimensional vector $\boldsymbol{\beta}_b$.

Initialization. The SfM [43] point cloud used in 3D Gaussian is suitable for object centric scene. However, it can not provide good initialization for urban street scenes with many under-observed or textureless regions. We instead use aggregated LiDAR point cloud captured by ego vehicle as initialization. The

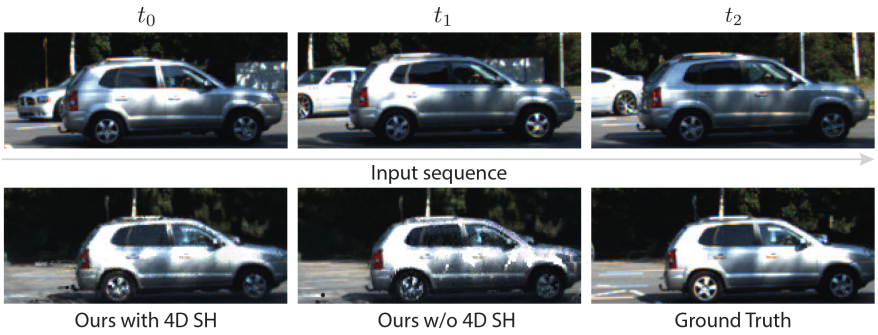

Fig. 3. Effect of 4D SH (spherical harmonics) model. The first row presents the input sequence, showcasing varying appearances. The second row demonstrates the impact of utilizing the proposed 4D SH model on the rendering results. Significant artifacts can be observed if the 4D SH model is absent.

colors of LiDAR point cloud are obtained by projecting to the corresponding image plane and querying the pixel value.

To initialize the object model, we first collect aggregated points inside the 3D bounding boxes and transform them into the local coordinate system. For object with less than 2K LiDAR points, we instead randomly sample 8K points inside the 3D bounding box as initialization. For the background model, we perform voxel downsampling for the remaining point cloud and filter out those that are invisible to the training cameras. We incorporate SfM point cloud to compensate for the limited coverage of LiDAR over large areas.

3.2 Rendering of Street Gaussians

To render Street Gaussians, we need to aggregate the contribution of each model to render the final image. Previous methods [22,37,60,67] require compositional rendering with complex raymarching because of neural field representation. Instead, Street Gaussians can be rendered by contacting all the point clouds and projecting them to 2D image space. Specifically, given a rendered time step t, we first compute spherical harmonics Eq. 3 , and transform the object point cloud into the world coordinate system using and Eq. 2 according to tracked vehicle pose $(\mathbf{R}_t, \mathbf{T}_t)$. Then we concatenate the background point cloud and the transformed object point clouds to form a new point cloud. To project this point cloud to 2D image space with camera extrinsic $\mathbf{W}$ and intrinsic $\mathbf{K}$, we compute the 2D Gaussian for each point in the point cloud [76]:

$$\begin{aligned} \boldsymbol{\mu}' &= \mathbf{KW}\boldsymbol{\mu}, \\ \boldsymbol{\Sigma}' &= \mathbf{JW}\boldsymbol{\Sigma}\mathbf{W}^T\mathbf{J}^T, \end{aligned} \tag{4}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathbf{K}$. μ' and Σ' are the position and covariance matrix in 2D image space, respectively. Point-based α-blending for each pixel is used to compute the color $\mathbf{C}$:

$$\mathbf{C} = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{5}$$

Here α_i is the opacity α multiplied by the probability of the 2D Gaussian and $\mathbf{c}_i$ is the color computed from spherical harmonics $\mathbf{z}$ with the view direction. We can also render other signals like depth, opacity and semantic. For instance, the semantic map is rendered by changing color c in Eq. 5 to semantic logits $\boldsymbol{\beta}$.

Since 3D Gaussian is defined in Euclidean space, it is inappropriate for them to model distant regions like sky. As a result, we utilize a high resolution cubemap which maps the view direction to sky color $\mathbf{C}_{\text{sky}}$. The explicit cubemap representation helps us recover details in sky regions without sacrificing inference speed. The final rendering color is obtained by blending $\mathbf{C}_{\text{sky}}$ and the color $\mathbf{C}$ in Eq. 5. More details can be found in the supplementary.

3.3 Training

Tracking Pose Optimization. Positions and covariance matrices of the object Gaussians during rendering in Sect. 3.2 are closely correlated with the tracked pose parameters as shown in Eq. 2. However, bounding boxes produced by the tracker model are generally noisy. Directly using them to optimize our scene representation leads to degradation in rendering quality. As a result, we treat tracked poses as learnable parameters by adding a learnable transformation to each transformation matrix. Specifically, $\mathbf{R}_t$ and $\mathbf{T}_t$ in Eq. 2 are replaced by $\mathbf{R}_t^{'}$ and $\mathbf{T}_t^{'}$ which are defined as:

$$\begin{aligned} \mathbf{R}_t' &= \mathbf{R}_t \Delta\mathbf{R}_t, \\ \mathbf{T}_t' &= \mathbf{T}_t + \Delta\mathbf{T}_t, \end{aligned} \tag{6}$$

where $\Delta\mathbf{R}_t$ and $\Delta\mathbf{T}_t$ are the learnable transformation. We represent $\Delta\mathbf{T}_t$ as a 3D vector and $\Delta\mathbf{R}_i$ as a rotation matrix converted from yaw offset angle $\Delta\theta_t$. Gradients of these transformations can be directly obtained without any implicit function or intermediate processes, which do not require any extra computation during back-propagation.

Loss Function. We jointly optimize our scene representation, sky cubemap and tracked poses using the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{color}} + \lambda_1 \mathcal{L}_{\text{depth}} + \lambda_2 \mathcal{L}_{\text{sky}} + \lambda_3 \mathcal{L}_{\text{sem}} + \lambda_4 \mathcal{L}_{\text{reg}}. \tag{7}$$

In Eq. 7, $\mathcal{L}_{\text{color}}$ is the reconstruction loss between rendered and observed images following [19]. $\mathcal{L}_{\text{depth}}$ is a L1 loss between rendered depth and the depth generated by projecting sparse LiDAR points onto the camera plane. $\mathcal{L}_{\text{sky}}$ is a binary cross entropy loss for sky supervision. $\mathcal{L}_{\text{sem}}$ is an optional per-pixel softmax-cross-entropy loss between rendered semantic logits and input 2D semantic segmentation predictions [23] and $\mathcal{L}_{\text{reg}}$ is an regularization term used to remove floaters and enhance decomposition effects. Please refer to the supplementary material for details of each loss term.

4 Implementation Details

We train Street Gaussians for 30000 iterations with Adam optimizers [20] following the configurations of 3D Gaussians [19]. The learning rate of translation transformation $\Delta\mathbf{T}_t$ and rotation transformation $\Delta\mathbf{R}_t$ are set to $5e^{-3}$ and $1e^{-3}$, which decay exponentially to $5e^{-5}$ and $1e^{-5}$ respectively. The resolution of sky cubemap is set to 1024 with learning rate decays from $1e^{-2}$ to $1e^{-4}$ exponentially. All the experiments are conducted on one single RTX 4090 GPU.

We follow [19] to apply adaptive control during optimization. We fix the scale of background model (20 m in our experiments) and the scale of each object model is determined by the bounding box dimensions. In order to prevent object Gaussians from growing to occluded areas, for each object model we sample a set of points as a probability distribution function. During optimization, Gaussians with sampled points outside the bounding box will be pruned.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on Waymo Open Dataset [47] and KITTI benchmarks [15]. The frame rates of both datasets are 10 HZ. On the Waymo Open Dataset, we select 8 recording sequences with large amounts of moving objects, significant ego-car motion and complex lighting conditions. All sequences have a length of around 100 frames. We select every 4th image in the sequence as the test frames and use the remaining for training. As we find that our baseline methods [37,60] suffer from high memory cost when training with high-resolution images, we downscale the input images to 1066×1600. On KITTI [15] and Vitural KITTI 2 [6], we follow the settings of MARS [60] and evaluate our methods with different train/test split settings. We use the bounding boxes generated by the detector [58] and tracker [59] on Waymo dataset and use the officially provided object tracklets from KITTI.

Baseline Methods. We compare our methods with four recent methods. (1) NSG [37] represents background as multi-plane images and use per-object learned latent codes with a shared decoder to model moving objects. (2) MARS [60] builds the neural scene graph based on Nerfstudio [49]. (3) 3D Gaussians [19] models the scene with a set of anisotropy gaussians. (4) EmerNeRF [65] stratifies scenes into static and dynamic fields, each modeled with a hash grid [34]. Both NSG and MARS are trained and evaluated using ground truth object tracklets. Details of baseline implementations can be found in the supplementary.

5.2 Comparisons with the State-of-the-Art

Tables 1, 2 present the comparison results of our method with baseline methods [19,37,60,65] in terms of rendering quality and rendering speed. We adopt PSNR, SSIM and LPIPS [73] as metrics to evaluate rendering quality. To better evaluate

Fig. 4. Qualitative comparisons results on the Waymo [47] dataset. NSG [37] and MARS [60] often produce blurry and distorted results. 3D GS [19] and EmerNeRF [65] generates ghosting artifacts in regions with moving objects. In contrast, our approach significantly outperforms other methods with high fidelity and sharp details.

Table 1. Quantitative results on the Waymo [47] dataset. The rendering image resolution is 1066 × 1600. "PSNR*" denotes the PSNR of moving objects.

	3D GS [19]	NSG [37]	MARS [60]	EmerNeRF [65]	Ours
PSNR↑	29.64	28.31	29.75	30.87	**34.61**
SSIM↑	0.918	0.862	0.886	0.905	**0.938**
LPIPS↓	0.117	0.346	0.264	0.133	**0.079**
PSNR*↑	21.25	24.32	26.54	21.67	**30.23**
FPS↑	**205**	0.47	0.68	0.21	135

Table 2. Quantitative results on KITTI [15] and VKITTI2 [6] datasets. We strictly follow the experimental setting of MARS [60] and borrow results of MARS [60] and NSG [37] from it. The rendering image resolution is 375 × 1242.

	KITTI - 75%			KITTI - 50%			KITTI - 25%		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D GS [19]	19.19	0.737	0.172	19.23	0.739	0.174	19.06	0.730	0.180
NSG* [37]	21.53	0.673	0.254	21.26	0.659	0.266	20.00	0.632	0.281
MARS* [60]	24.23	**0.845**	0.160	24.00	0.801	0.164	23.23	0.756	0.177
Ours	**25.79**	0.844	**0.081**	**25.52**	**0.841**	**0.084**	**24.53**	**0.824**	**0.090**
	VKITTI2 - 75%			VKITTI2 - 50%			VKITTI2 - 25%		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3D GS [19]	21.12	0.877	0.097	21.11	0.874	0.097	20.84	0.863	0.098
NSG* [37]	23.41	0.689	0.317	23.23	0.679	0.325	21.29	0.666	0.317
MARS* [60]	29.79	0.917	0.088	29.63	0.916	0.087	27.01	0.887	0.104
Ours	**30.10**	**0.935**	**0.025**	**29.91**	**0.932**	**0.026**	**28.52**	**0.917**	**0.034**

the rendering quality of moving objects, we project 3D bounding boxes to 2D image plane and calculate the loss only on pixels inside the projected box, which is denoted as PSNR* in our experiments. For all the metrics, our model achieves the best performance among all the methods with a 12.1% increase in PSNR and a 13.9% increase in PSNR*. Moreover, our method renders two magnitudes faster than NeRF-based methods [37,60,65]. Although 3D GS is faster than our method, it can only support static scenes and the result of moving objects degrades significantly.

Figure 4 shows the qualitative results of our method and baselines on the Waymo dataset. 3D GS fails to model dynamic objects and EmerNeRF can not generates reasonable results in dynamic regions of novel timestep. Although given ground truth tracking poses, NSG and MARS still suffer from blurry and distorted results due to the lack of capacity of their model when the scene is complex. In contrast, our method can generate high-quality novel views with high fidelity and details.

Table 3. Ablation studies on the Waymo [47] dataset. Metrics are averaged over all the sequences on the Waymo dataset. "PSNR*" denotes the PSNR of moving objects. "opt." denotes optimization. Please refer to Sect. 5.3 for details.

	PSNR↑	PSNR*↑	SSIM↑	LPIPS↓
Ours w/o LiDAR	34.02	29.53	0.934	0.087
Ours w/o 4DSH	34.36	29.27	0.937	0.081
Ours w/o pose opt.	34.18	28.24	0.935	0.081
Ours w/ GT pose	**34.61**	29.84	0.937	0.080
Complete model	**34.61**	**30.23**	**0.938**	**0.079**

Fig. 5. Ablation study on tracking pose optimization. The results indicate that optimizing tracked poses improves the quality. "opt." denotes optimization.

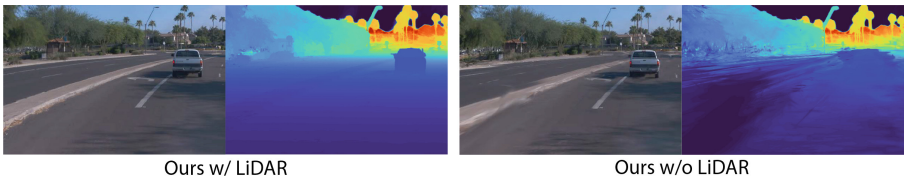

Fig. 6. Ablation study on LiDAR point cloud. We show the rendered image and depth of our method with and without LiDAR as input.

5.3 Ablations and Analysis

We validate our algorithm's design choices on all selected sequences from the Waymo dataset. Table 3 presents the quantitative results.

Importance of Optimizing Tracked Poses. Experimental results in Table 3 show that our complete model outperforms the model trained without tracking pose optimization by a large margin, which indicates the effectiveness of our pose optimization strategy. It is interesting to notice that the result of our method is even better than the model trained with ground truth poses, a plausible explanation is that there still exists noise in ground truth annotations.

Visual results of the influence of tracked pose optimization is shown in Fig. 5. Treating tracked poses as learnable parameters help the object model synthesize more texture details like the rear of the white vehicle or the logo of the black vehicle and reduce rendering artifacts.

Effectiveness of 4D Spherical Harmonics. Results in Table 3 indicate that our 4D spherical harmonics appearance model can refine the rendering quality. This situation becomes particularly evident when the object interacts with environmental lighting as shown in Fig. 3. Our model can generate smooth shadows on the car while the rendering results without 4D spherical harmonics are much noisier.

Influence of Incorporating LiDAR Points. We evaluate the influence of LiDAR point cloud by comparing our method to a variant with SfM initialization for background and random initialization for moving object as described

Fig. 7. Editing operations on the Waymo [47] dataset. Images in the first and second rows represent the results before and after editing. Our method supports various editing operations, including rotation, translation and swapping.

Fig. 8. Decomposition results on the Waymo [47] dataset. NSG [37] cannot decompose clean foreground objects while MARS [60] generates floaters in background regions. Instead, our method successfully decomposes the foreground objects and produces high fidelity rendering results.

Table 4. Quantitative segmentation results on the KITTI [15] dataset. "VKN ground-truth" and "VKN rendered" denote semantic prediction results of Video K-Net with ground-truth images and our rendered images, respectively.

Method	VKN ground-truth	VKN rendered	Ours
mIoU ↑	57.94	53.81	**58.81**

in Sect. 3.1. We also disable the LiDAR depth loss in Eq. 7. Table 3 shows that incorporating LiDAR point cloud enhances the results of both background and moving objects. Figure 6 indicates that using LiDAR points helps our model recover more accurate scene geometry and reduce blurry artifacts. It is worth noticing that our method still significantly outperforms baseline methods even without LiDAR input as shown in Tables 3, 4, which proves the efficiency of Street Gaussians under different settings.

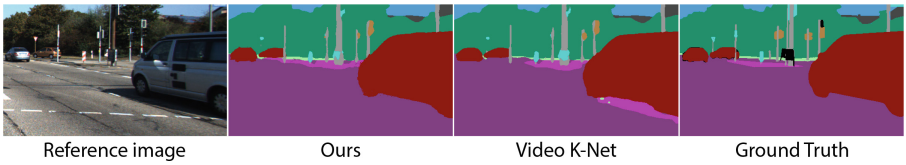

Fig. 9. Visual semantic segmentation results on the KITTI [15] dataset. It can be observed that our method achieves better performance, particularly in ambiguous areas such as shadows, due to our ability to fuse semantic information in 3D.

5.4 Applications

Street Gaussians can be applied to multiple tasks in computer vision including object decomposition, semantic segmentation and scene editing.

Scene Editing. Our instance-aware scene representation enables various types of scene editing operations. We can rotate the heading of the vehicle (Fig. 7(a)), translate the vehicle (Fig. 7(b)) and swap one vehicle in the scene with another one (Fig. 7(c)).

Object Decomposition. We compare the decomposition results of our method with NSG [37] and MARS [60] under the Waymo dataset. As shown in Fig. 8, NSG fails to disentangle foreground objects from the background and the result of MARS is blurry due to the model capacity and lack of regularization. In contrast, our method can produce high-fidelity and clean decomposed results.

Semantic Segmentation. We compare the quality of our rendered semantic map with the semantic prediction from Video-K-Net [23] on KITTI dataset. Our semantic segmentation model is trained with results from Video K-Net. Qualitative and quantitative results are shown in Fig. 9 and Table 4. Our semantic maps achieve better performance thanks to our representation.

6 Conclusion

This paper introduced Street Gaussians, an explicit scene representation for modeling dynamic urban street scenes. The proposed representation separately models the background and foreground vehicles as a set of neural point clouds. This explicit representation allows easy compositing of object vehicles and background, enabling scene editing and real-time rendering within half an hour of training. Furthermore, we demonstrate that the proposed scene representation can achieve comparable performance to that achieved using precise ground-truth poses, using only poses from an off-the-shelf tracker. Detailed ablation and comparison experiments are conducted on several datasets, demonstrating the effectiveness of the proposed method.

Acknowledgement. The authors would like to acknowledge the support from NSFC (No. 623B2091), Li Auto and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

1. Aliev, K.-A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12367, pp. 696–712. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58542-6_42
2. Attal, B., et al.: HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In: CVPR (2023)
3. 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: ICCV (2021)
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: unbounded anti-aliased neural radiance fields. In: CVPR (2022)
5. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: anti-aliased grid-based neural radiance fields. In: ICCV (2023)
6. Cabon, Y., Murray, N., Humenberger, M.: Virtual kitti 2. arXiv preprint arXiv:2001.10773 (2020)
7. Chen, Y., et al.: GeoSim: realistic video simulation via geometry-aware composition for self-driving. In: CVPR (2021)
8. Chen, Y., Gu, C., Jiang, J., Zhu, X., Zhang, L.: Periodic vibration Gaussian: dynamic urban scene reconstruction and real-time rendering. arXiv:2311.18561 (2023)
9. Cheng, K., et al.: Uc-NeRF: neural radiance field for under-calibrated multi-view cameras. In: ICLR (2024)
10. Dai, P., Zhang, Y., Li, Z., Liu, S., Zeng, B.: Neural point cloud rendering via multi-plane projection. In: CVPR (2020)
11. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: an open urban driving simulator. In: CoRL (2017)
12. Fang, J., et al.: Augmented lidar simulator for autonomous driving. IEEE Robot. Autom. Lett. **5**(2), 1931–1938 (2020)
13. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: explicit radiance fields in space, time, and appearance. In: CVPR (2023)
14. Fu, X., et al.: Panoptic NeRF: 3D-to-2D label transfer for panoptic urban scene segmentation. In: 3DV (2022)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
16. Guo, J., et al.: StreetSurf: extending multi-view implicit surface reconstruction to street views. arXiv preprint arXiv:2306.04988 (2023)
17. Huang, S., et al.: Neural lidar fields for novel view synthesis. In: ICCV (2023)
18. Irshad, M.Z., et al.: Neo 360: neural fields for sparse view synthesis of outdoor scenes. In: ICCV (2023)
19. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. TOG **42**(4) (2023)
20. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Kopanas, G., Philip, J., Leimkühler, T., Drettakis, G.: Point-based neural rendering with per-view optimization. In: CGF, vol. 40, pp. 29–43. Wiley Online Library (2021)
22. Kundu, A., et al.: Panoptic neural fields: a semantic object-aware neural scene representation. In: CVPR (2022)

23. Li, X., et al.: Video k-net: a simple, strong, and unified baseline for video segmentation. In: CVPR (2022)
24. Li, Y., Lin, Z.H., Forsyth, D., Huang, J.B., Wang, S.: ClimateNeRF: physically-based neural rendering for extreme climate synthesis. In: ICCV (2023)
25. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR (2021)
26. Li, Z., Li, L., Zhu, J.: Read: large-scale neural scene rendering for autonomous driving. In: AAAI (2023)
27. Lin, H., et al.: High-fidelity and real-time novel view synthesis for dynamic scenes. In: SIGGRAPH (2023)
28. Lin, H., et al.: Efficient neural radiance fields for interactive free-viewpoint video. In: SIGGRAPH (2022)
29. Liu, J.Y., Chen, Y., Yang, Z., Wang, J., Manivasagam, S., Urtasun, R.: Neural scene rasterization for large scene rendering in real time. In: ICCV (2023)
30. Lu, F., Xu, Y., Chen, G., Li, H., Lin, K.Y., Jiang, C.: Urban radiance field representation with deformable neural mesh primitives. In: ICCV (2023)
31. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3D Gaussians: tracking by persistent dynamic view synthesis. In: 3DV (2024)
32. Manivasagam, S., et al.: LiDARsim: realistic lidar simulation by leveraging the real world. In: CVPR (2020)
33. 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.) ECCV 2020. LNCS, vol. 12346, pp. 405–421. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_24
34. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. In: SIGGRAPH (2022)
35. Niemeyer, M., Geiger, A.: Giraffe: representing scenes as compositional generative neural feature fields. In: CVPR (2021)
36. Ost, J., Laradji, I., Newell, A., Bahat, Y., Heide, F.: Neural point light fields. In: CVPR (2022)
37. Ost, J., Mannan, F., Thuerey, N., Knodt, J., Heide, F.: Neural scene graphs for dynamic scenes. In: CVPR (2021)
38. Park, K., et al.: HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. TOG **40**(6) (2021)
39. Peng, S., Yan, Y., Shuai, Q., Bao, H., Zhou, X.: Representing volumetric videos as dynamic MLP maps. In: CVPR (2023)
40. Pun, A., et al.: Neural lighting simulation for urban scenes. In: NeurIPS (2023)
41. Rematas, K., et al.: Urban radiance fields. In: CVPR (2022)
42. Rückert, D., Franke, L., Stamminger, M.: ADOP: approximate differentiable one-pixel point rendering. TOG **41**(4), 1–14 (2022)
43. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
44. Shah, S., Dey, D., Lovett, C., Kapoor, A.: AirSim: high-fidelity visual and physical simulation for autonomous vehicles. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics. SPAR, vol. 5, pp. 621–635. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67361-5_40
45. Shen, B., et al.: GINA-3D: learning to generate implicit neural assets in the wild. In: CVPR (2023)
46. Song, L., et al.: NeRFPlayer: a streamable dynamic scene representation with decomposed neural radiance fields. TVCG **29**(5), 2732–2742 (2023)

47. Sun, P., et al.: Scalability in perception for autonomous driving: waymo open dataset. In: CVPR (2020)
48. Tancik, M., et al.: Block-nerf: scalable large scene neural view synthesis. In: CVPR (2022)
49. Tancik, M., et al.: Nerfstudio: a modular framework for neural radiance field development. In: SIGGRAPH 2023 Conference Proceedings (2023)
50. Tonderski, A., Lindström, C., Hess, G., Ljungbergh, W., Svensson, L., Petersson, C.: NeuRAD: neural rendering for autonomous driving. In: CVPR (2024)
51. Turki, H., Ramanan, D., Satyanarayanan, M.: Mega-NeRF: scalable construction of large-scale nerfs for virtual fly-throughs. In: CVPR (2022)
52. Turki, H., Zhang, J.Y., Ferroni, F., Ramanan, D.: SUDS: scalable urban dynamic scenes. In: CVPR (2023)
53. Wang, J., et al.: CADSim: robust and scalable in-the-wild 3D reconstruction for controllable sensor simulation. In: CoRL (2022)
54. Wang, Z., Chen, W., Acuna, D., Kautz, J., Fidler, S.: Neural light field estimation for street scenes with differentiable virtual object insertion. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13662, pp. 380–397. Springer, Cham (2022)
55. Wang, Z., et al.: Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In: CVPR (2023)
56. Wei, Y., et al.: Editable scene simulation for autonomous driving via collaborative llm-agents. In: CVPR (2024)
57. Wu, G., et al.: 4D Gaussian splatting for real-time dynamic scene rendering. In: CVPR (2024)
58. Wu, H., Deng, J., Wen, C., Li, X., Wang, C.: Casa: a cascade attention network for 3D object detection from lidar point clouds. IEEE Trans. Geosci. Remote Sens. (2022)
59. Wu, H., Han, W., Wen, C., Li, X., Wang, C.: 3D multi-object tracking in point clouds based on prediction confidence-guided data association. IEEE Trans. Intell. Transp. Syst. **23**(6), 5668–5677 (2021)
60. Wu, Z., et al.: Mars: an instance-aware, modular and realistic simulator for autonomous driving. In: CICAI (2023)
61. Xie, Z., Zhang, J., Li, W., Zhang, F., Zhang, L.: S-Nerf: neural radiance fields for street views. In: ICLR (2023)
62. Xu, L., et al.: Grid-guided neural radiance fields for large urban scenes. In: CVPR (2023)
63. Xu, Y., et al.: Discoscene: spatially disentangled generative radiance fields for controllable 3D-aware scene synthesis. In: CVPR (2023)
64. Xu, Z., et al.: 4K4D: Real-time 4D view synthesis at 4K resolution. In: CVPR (2024)
65. Yang, J., et al.: EmerNeRF: emergent spatial-temporal scene decomposition via self-supervision. In: ICLR (2024)
66. Yang, Y., Yang, Y., Guo, H., Xiong, R., Wang, Y., Liao, Y.: Urbangiraffe: representing urban scenes as compositional generative neural feature fields. In: ICCV (2023)
67. Yang, Z., et al.: UniSim: a neural closed-loop sensor simulator. In: CVPR (2023)
68. Yang, Z., Manivasagam, S., Chen, Y., Wang, J., Hu, R., Urtasun, R.: Reconstructing objects in-the-wild for realistic sensor simulation. In: ICRA (2023)
69. Yang, Z., Yang, H., Pan, Z., Zhu, X., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: ICLR (2024)

70. Yang, Z., et al.: SurfelGAN: synthesizing realistic sensor data for autonomous driving. In: CVPR (2020)
71. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3D gaussians for high-fidelity monocular dynamic scene reconstruction. In: CVPR (2024)
72. Zhang, Q., Baek, S.H., Rusinkiewicz, S., Heide, F.: Differentiable point-based radiance fields for efficient view synthesis. In: SIGGRAPH, pp. 1–12 (2022)
73. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
74. Zhang, X., Kundu, A., Funkhouser, T., Guibas, L., Su, H., Genova, K.: Nerflets: local radiance fields for efficient structure-aware 3D scene representation from 2D supervision. In: CVPR (2023)
75. Zhou, X., Lin, Z., Shan, X., Wang, Y., Sun, D., Yang, M.H.: DrivingGaussian: composite gaussian splatting for surrounding dynamic autonomous driving scenes. arXiv preprint arXiv:2312.07920 (2023)
76. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: EWA volume splatting. In: Proceedings Visualization, 2001. VIS 2001, pp. 29–538. IEEE (2001)