UniGaussian: Driving Scene Reconstruction from Multiple Camera Models via Unified Gaussian Representations

Yuan Ren[1], Guile Wu[1], Runhao Li[1,2,*], Zheyuan Yang[1,2,*], Yibo Liu[1,3,*]
Xingxin Chen[1], Tongtong Cao[1], and Bingbing Liu[1,†]
[1]Huawei Noah's Ark Lab [2]University of Toronto [3]York University
{yuan.ren3, xingxin.chen1, caotongtong, liu.bingbing}@huawei.com, guile.wu@outlook.com
{rain.li, andrewzheyuan.yang}@mail.utoronto.ca, buaayorklau@gmail.com

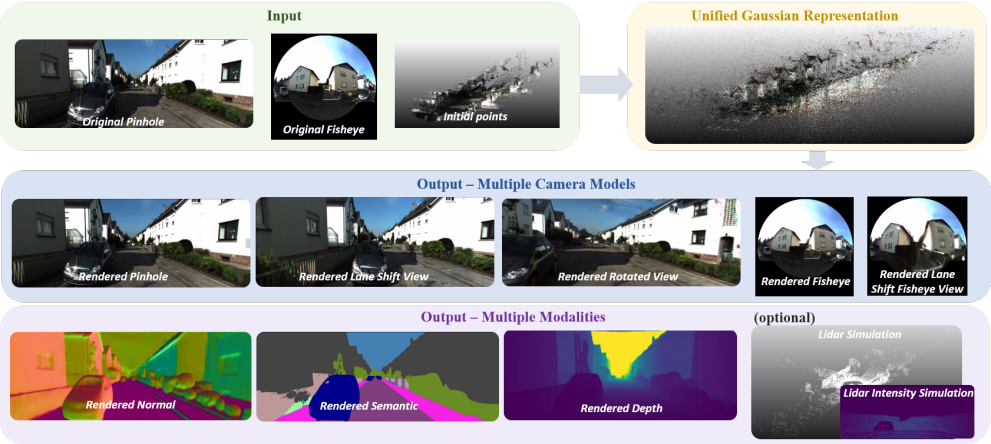


Figure 1. An illustration of the proposed approach. UniGaussian reconstructs 3D driving scenes by learning unified 3D Gaussian representations from multiple input sources. It achieves holistic driving scene understanding and models multiple sensors (pinhole cameras and fisheye cameras) and modalities (semantic, normal, depth, and optional LiDAR point clouds).

Abstract

Urban scene reconstruction is crucial for real-world autonomous driving simulators. Although existing methods have achieved photorealistic reconstruction, they mostly focus on pinhole cameras and neglect fisheye cameras. In fact, how to effectively simulate fisheye cameras in driving scenes remains an unsolved problem. In this work, we propose UniGaussian, a novel approach that learns unified 3D Gaussian representations from multiple camera models for urban scene reconstruction in autonomous driving. Our contributions are two-fold. First, we propose a new differentiable rendering method that distorts 3D Gaussians using a series of affine transformations tailored to fisheye camera models. This addresses the compatibility issue of 3D Gaussian splatting with fisheye cameras, which is hindered by light ray distortion caused by lenses or mirrors. Besides, our method maintains real-time rendering while ensuring differentiability. Second, built on the differentiable rendering method, we design a new framework that learns unified Gaussian representations from multiple camera models. By applying affine transformations to adapt different camera models and regularizing the shared Gaussians with supervision from different modalities, our framework learns unified 3D Gaussian representations with input data from multiple sources and achieves holistic driving scene understanding. As a result, our approach models multiple sensors (pinhole and fisheye cameras) and modalities (depth, semantic, normal, and LiDAR point clouds). Our experiments show that our method achieves superior rendering quality and fast rendering speed for driving scene simulation.

1. Introduction

Urban scene reconstruction aims to reconstruct real-world driving scenes from images and achieve photorealistic rendering [3, 26, 33, 36]. It is crucial for realizing real-world closed-loop evaluation in end-to-end autonomous driving

[*]R. Li, Z. Yang, and Y. Liu made contributions during their internships at Huawei Canada. [†]B. Liu is the corresponding author.

simulators [31, 33]. One of the most popular methods for driving scene reconstruction is the use of Neural Radiance Field (NeRF) [21], an implicit radiance field representation. Although NeRF-based methods [8, 23, 26, 29, 33] have achieved photorealistic rendering, they suffer from slow rendering speed and expensive training cost. Recently, some researchers have resorted to 3D Gaussian Splatting (3DGS) [11], an explicit radiance field representation, for urban scene reconstruction. 3DGS-based methods [3, 32, 36, 37] use explicit 3D Gaussian primitives to represent 3D scenes and employ a differentiable tile rasterizer for rendering. This enhances the editability of driving scenes and achieves real-time photorealistic rendering.

However, existing driving scene reconstruction methods mainly focus on rendering from pinhole cameras and largely neglect fisheye cameras. Compared with pinhole cameras, fisheye cameras provide a wider field of view (FOV) essential for navigation and perception tasks in autonomous driving, especially for near-field sensing and automatic parking [14, 15, 24]. Nevertheless, it is non-trivial to adapt 3DGS to fisheye cameras. For example, the direct application of 3DGS to fisheye cameras is impeded by distortions induced by camera optics, which disrupt the affine transformations of 3D Gaussians. Intuitively, an engineering strategy to tackle this issue is directly rectifying fisheye images and treating them as pinhole images for model training. However, this strategy yields suboptimal rendering quality, particularly in regions with a large field of view angle. Another strategy is to render multiple cubic images and subsequently amalgamate them to construct fisheye images [13, 27]. Nevertheless, this strategy introduces noticeable artifacts indicative of the stitching process. In fact, how to effectively render driving scenes from fisheye images for simulation remains an open question.

To address the aforementioned challenges, this work presents UniGaussian, a novel 3DGS-based approach that learns unified Gaussian representations from multiple camera models for urban scene reconstruction in autonomous driving. First, we aim to address the compatibility issue of 3DGS with fisheye cameras. To this end, we propose a novel differentiable rendering method for 3DGS tailored to fisheye cameras. In our method, the distortion of light rays caused by fisheye cameras is translated into the deformation of the radiance field. This is realized through a series of affine transformations, including translation, rotation, and stretching, applied to 3D Gaussians. These operations maintain the parallelization strategy and real-time performance of 3DGS, while ensuring differentiability. Second, with the proposed differentiable rendering method, we design a new framework that jointly optimizes unified 3D Gaussian representations from multiple camera models for driving scene reconstruction. We apply affine transformations on 3D Gaussians to adapt them for different camera models and regularize the shared Gaussians with supervision from different modalities. In this way, our framework learns unified 3D Gaussian representations with input data from multiple sources and achieves holistic driving scene understanding. As illustrated in Fig. 1, our approach models multiple sensors, including pinhole cameras and fisheye cameras, and multiple modalities, including depth, semantic, normal, and optional LiDAR point clouds. Besides, since we use a unified rasterizer for 3D Gaussian rendering, our framework shows adaptability to various differentiable camera models and maintains real-time performance. The framework of the proposed approach is depicted in Fig. 2.

In summary, our **contributions** are two-fold:

- We propose a novel differentiable rendering method for 3DGS tailored to driving scene reconstruction from fisheye cameras. Through a series of affine transformations, our method has adaptability to various fisheye camera models for driving scene reconstruction with large FOVs, does not significantly increase GPU memory consumption and maintains real-time rendering.
- Built on the proposed differentiable rendering method, we propose to learn unified Gaussian representations from multiple camera models for driving scene reconstruction. Our approach achieves holistic driving scene understanding by modeling multiple sensors and modalities. To the best of our knowledge, no existing driving scene reconstruction method simulates both pinhole and fisheye cameras in a unified framework.

2. Related Work

3D Gaussian Splatting. 3DGS [11] is a real-time radiance field rendering approach. It represents a 3D scene with a set of explicit 3D Gaussian primitives. Each primitive is defined by a position (mean) $\mu \in \mathbb{R}^3$, a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, an opacity $o \in \mathbb{R}^1$ and spherical harmonics (colors). To render images, these 3D Gaussian primitives are projected from 3D space onto 2D image plane to compute the color for each pixel with a differentiable tile-based rasterizer. 3DGS achieves highly effective training and photorealistic real-time rendering. We refer readers to [11] for more details of 3DGS.

With the incredible success achieved by 3DGS, many 3DGS-based methods have emerged with the aim of solving the problems of anti-aliasing [35], acceleration [6], relighting [7], sparse-view synthesis [30], 4D reconstruction [28], *etc*. Most of these works focus on reconstruction with pinhole camera models only. Recently, Liao *et al*. [17] adapt 3DGS to fisheye cameras by recalculating 3DGS projection and gradients, but as they point out, their method is only based on ideal camera models with equidistant projection and not for generic fisheye camera models and real large-FOV cameras. This hinders their use in driving scene reconstruction because fisheye cameras in driving scene are

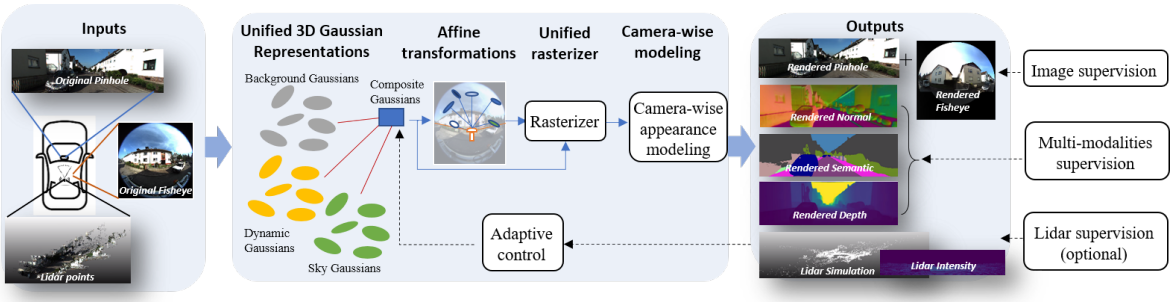


Figure 2. The framework of our UniGaussian approach to driving scene reconstruction with multiple camera models. Our approach achieves holistic driving scene understanding by modeling multiple sensors and modalities.

usually generic models and have large FOVs. In contrast, our work proposes a new differentiable rendering method for 3DGS tailored to driving scene reconstruction with fisheye cameras. Through a series of affine transformations, our method has adaptability to various fisheye camera models for driving scene reconstruction with large FOVs, does not significantly increase GPU memory consumption and maintains real-time rendering.

Driving Scene Reconstruction. Typically, autonomous driving scene simulation involves driving scene reconstruction [26, 29, 33] and road asset reconstruction [18, 19]. Our work mainly focuses on driving scene reconstruction from images. Contemporary driving scene reconstruction methods can be mainly categorized into two types, namely NeRF-based methods and 3DGS-based methods. Generally, NeRF-based methods [8, 23, 26, 29, 33] perform dense ray sampling and leverage a multi-layer perception with many fully-connected layers to represent 3D scenes. Although they have shown promising results for photorealistic rendering, they are usually slow for both training and rendering. This hinders their application to real-world autonomous driving simulators. On the other hand, 3DGS-based methods [3, 32, 36, 37] appear to be more efficient for driving scene reconstruction due to the efficiency of explicit 3D Gaussian representations and differentiable rasterization. They are able to achieve photorealistic reconstruction and real-time rendering. Nevertheless, existing NeRF-based and 3DGS-based methods neglect the importance of fisheye camera simulation for autonomous driving. In autonomous driving, fisheye cameras are useful for providing a wider field of view, especially for near-field sensing and automatic parking. Our work differs from existing works in that the proposed method is built on a novel differentiable rendering method tailored to fisheye cameras and the proposed framework learns unified 3D Gaussians by modeling multiple sensors and modalities.

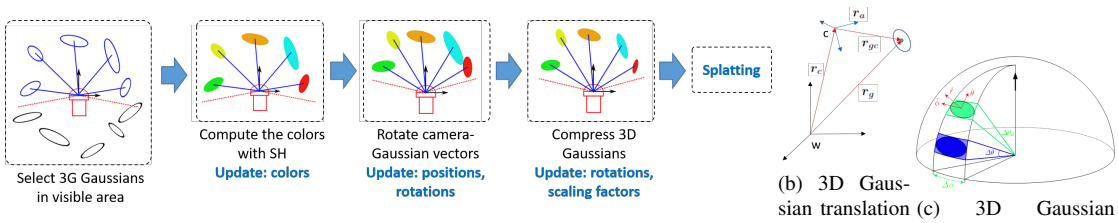
Although we use composite Gaussians as the driving scene representation following [32, 36, 37], our approach significantly differs from [32, 36, 37]. Our approach presents a novel differentiable rendering method tailored to fisheye cameras and constructs a unified framework to learn unified 3D Gaussians by modeling multiple sensors and modalities. In contrast, [32, 36, 37] are designed to model driving scene with pinhole cameras only. Specifically, [32] constructs composite street Gaussians as driving scene representation and employs tracking pose optimization to improve foreground object modeling; [36] employs composite Gaussians for holistic scene understanding with a unicycle model for foreground object modeling; [37] presents incremental 3D Gaussians and dynamic Gaussian graph for driving scene reconstruction. None of them simulates multiple camera models in a unified framework for driving scene reconstruction. Note that post-processing or redistrotion is not a practical solution to 3DGS fisheye rendering in driving scene reconstruction, which yields suboptimal rendering quality in regions with a large FOV (see Sec. 4.2 for comparison with 3DGS+Undistort) and prevents each camera model from learning complementary information.

3. Methodology

This work focuses on driving scene reconstruction from multiple camera models, including pinhole and fisheye cameras. Before delving into driving scene reconstruction from multiple camera models, it is necessary to discuss differentiable rendering of 3DGS for fisheye cameras. Thus, in Sec. 3.1, we briefly introduce mathematical models of fisheye cameras; then, in Sec. 3.2, we present the proposed differentiable rendering of 3DGS for fisheye cameras; finally, in Sec. 3.3, we introduce the proposed UniGaussian framework for driving scene reconstruction.

3.1. Mathematical Models of Fisheye Cameras

The Kannala-Brandt [10] and MEI [20] mathematical models are two models commonly used for fisheye cameras. Each model consists of two main steps. The first step can be referred to as the "mirror transformation" that characterizes the twisting of light rays induced by the lens/mirrors or the transformation that translates 3D points to new positions by rotating the camera-point vectors towards the optical axis of the camera. The second step is projecting 3D points from



(a) The architecture of our differentiable rendering for fisheye cameras. (b) 3D Gaussian translation & rotation. (c) 3D Gaussian stretching.

Figure 3. Illustrations of our 3DGS rendering with fisheye cameras.

new positions onto the image plane. This step utilizes either the camera intrinsic matrix or a pseudo intrinsic matrix.

Let θ and θ_d be the included angles between the camera-point vectors and the optical axis before and after applying the rotation, respectively. Then, the first step is defined as:

$$\theta_d = \mathcal{M}(\theta), \tag{1}$$

where $\mathcal{M}$ is the transformation specified by different models. For the Kannala-Brandt model, Eq. (1) is defined as:

$$\theta_d = \arctan r_d = \arctan\left(\theta\left(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8\right)\right), \tag{2}$$

where k_i is a distortion coefficient. For the MEI model, Eq. (1) is defined as:

$$\theta_d = \arctan r_d = \arctan\left(\chi + k_1\chi^3 + k_2\chi^5\right), \tag{3}$$

where $\chi = \frac{\sin\theta}{\cos\theta+\xi}$ and the twist of light rays is controlled by three parameters k_1, k_2 and ξ. Next, these translated 3D points are projected onto the image plane with intrinsic parameters f_x, f_y, u_0 and v_0. Specifically, the pseudo intrinsic matrix defined in Eq. (4) is used to project the translated 3D points onto the image plane. Since the focal length f_x, f_y and the gain η cannot be estimated independently, γ_1 and γ_2 are used as the pseudo focal length of the MEI model.

$$\begin{bmatrix} \gamma_1 & 0 & u_0 \\ 0 & \gamma_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x\eta & 0 & u_0 \\ 0 & f_y\eta & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

Note that the MEI model also considers the tangential distortion caused by the misalignment between the camera optical axis and the mirrors/lens rotational axis [20], but the influence of this distortion is minor so we neglect it here.

Discussion. From Eqs. (2) and (3), we can see that the principles of these two camera models are similar. We can move 3D points by rotating the camera-point vectors towards the optical axis of the camera and then project these 3D points onto the image plane with the pinhole camera model. From this perspective, the rendering of a scene represented by 3D Gaussians follows a similar procedure, with the main differences being that 3D points are replaced by 3D Gaussians and the rotational transformation of the camera-point vectors introduces the deformation to 3D Gaussians. In light of this, we formulate a transformation method to approximate the deformation of 3D Gaussians for fisheye cameras while maintaining differentiability.

3.2. 3DGS Rendering with Fisheye Cameras

Overview. The architecture of the proposed differentiable rendering method is shown in Fig. 3a. Suppose a driving scene is represented by a set of 3D Gaussians. When rendering from a specific viewpoint, we first select all visible 3D Gaussians and compute their colors utilizing spherical harmonics in the world space. Subsequently, we adjust the positions of the selected 3D Gaussians via rotating the camera-Gaussian center vectors towards the optical axis of the camera. Also, the poses of the selected 3D Gaussian are rotated with the camera-Gaussian center vectors. Then, the adjusted 3D Gaussians are compressed in both polar and tangential directions. The scaling factor and the quaternion of the 3D Gaussians are updated via eigendecomposition. Finally, the updated 3D Gaussians are projected onto the image plane to generate the final rendering. Overall, our method is generic to various differentiable camera models and can be used for driving scene reconstruction with large FOVs. Besides, our method does not require large amounts of additional GPU memory, so is suitable for processing millions of 3D Gaussians in driving scene reconstruction.

3D Gaussian Position and Pose Adjustment. Let $\boldsymbol{r}_g$ and $\boldsymbol{r}_c$ be the positions of 3D Gaussians and cameras in the world coordinate space, respectively, and let $\boldsymbol{r}_a$ be the optical axis of cameras. As shown in Fig. 3b, when performing the transformation, the camera-Gaussian center vector needs to be rotated by θ_Δ around the vector $\boldsymbol{r}_{rot}$. Here, $\theta_\Delta = \theta_d - \theta$, and $\boldsymbol{r}_{rot}$ is denoted by:

$$\boldsymbol{r}_{rot} = \frac{\boldsymbol{r}_{gc}}{||\boldsymbol{r}_{gc}||} \times \boldsymbol{r}_a, \tag{5}$$

where $\boldsymbol{r}_{gc} = \boldsymbol{r}_g - \boldsymbol{r}_c$, θ is the included angle between the camera-Gaussian center vector and $\boldsymbol{r}_a$, and θ_d is computed using Eq. (1). Then, the new position $\boldsymbol{r}'_g$ of the 3D Gaussian is computed as:

$$\boldsymbol{r}'_g = \boldsymbol{C}(\delta q)\boldsymbol{r}_{gc} + \boldsymbol{r}_c, \quad \text{where } \delta q = q_{vec}(\boldsymbol{r}_{rot}, \theta_\Delta), \tag{6}$$

and q_{vec} denotes the transform from the axis-angle to the quaternion and $\boldsymbol{C}(\cdot)$ is the transform from the quaternion to

the rotation matrix. Since the pose of the 3D Gaussian needs to be rotated along the camera-Gaussian center vector, the new quaternion q'_g is defined as:

$$q'_g = \delta q \otimes q_g, \tag{7}$$

where q_g is the original quaternion and $\otimes$ denotes the quaternion multiplication.

Compression of 3D Gaussians. Corresponding to the compression of FOV, 3D Gaussians need to be compressed in the polar and tangential directions; otherwise, adjacent 3D Gaussians will have greater degree of overlap on the splatting plane. To compress 3D Gaussians in the given direction, we use a stretching matrix $\boldsymbol{S}$ which is defined as:

$$\boldsymbol{S}(\hat{\boldsymbol{n}}, k) = \begin{bmatrix} 1+(k-1)n_x^2, & (k-1)n_x n_y, & (k-1)n_x n_z \\ (k-1)n_x n_y, & 1+(k-1)n_y^2, & (k-1)n_y n_z \\ (k-1)n_x n_z, & (k-1)n_y n_z, & 1+(k-1)n_z^2 \end{bmatrix}, \tag{8}$$

where $\hat{\boldsymbol{n}}$ is the stretching direction and k is the stretching ratio. This stretching matrix $\boldsymbol{S}$ is used to scale targets along an arbitrary axis [5]. In Fig. 3c, the blue and green ellipsoids denote 3D Gaussians before and after translation, rotation, and stretching. These two ellipsoids have the same tangential field angle $\Delta\phi$. The tangential stretching ratio is computed by $\sin\theta_d / \sin\theta$ and the polar stretching ratio is defined as:

$$k_\theta = \Delta\theta_d / \Delta\theta, \tag{9}$$

where $\Delta\theta = \max(||\hat{\theta} \cdot \mathbf{v}_x||, ||\hat{\theta} \cdot \mathbf{v}_y||, ||\hat{\theta} \cdot \mathbf{v}_z||) \times 2$, $\hat{\theta}$ is the local orthogonal unit vector points to the direction of optical axis, $\mathbf{v}_x$, $\mathbf{v}_y$ and $\mathbf{v}_z$ are the principal axes of the 3D Gaussian with scale. $\Delta\theta_d$ can be computed with the Taylor series expansion of Eq. (1) as:

$$\Delta\theta_d = \frac{d\theta_d}{d\theta}\Delta\theta + \frac{1}{2!}\frac{d^2\theta_d}{d\theta^2}\Delta\theta^2 + \frac{1}{3!}\frac{d^3\theta_d}{d\theta^3}\Delta\theta^3 + \cdots. \tag{10}$$

Therefore, for the Kannala-Brandt model, we have:

$$\frac{d\theta_d}{d\theta} = \frac{1}{1+r_d^2}\frac{dr_d}{d\theta}, \tag{11}$$

$$\frac{d^2\theta_d}{d\theta^2} = \frac{\frac{d^2 r_d}{d\theta^2}(1+r_d^2) - 2r_d(\frac{dr_d}{d\theta})^2}{(1+r_d^2)^2}, \tag{12}$$

where $\frac{dr_d}{d\theta} = 1 + 3k_1\theta^2 + 5k_2\theta^4 + 7k_3\theta^6 + 9k_4\theta^8$ and $\frac{d^2 r_d}{d\theta^2} = 6k_1\theta + 20k_2\theta^3 + 42k_3\theta^5 + 72k_4\theta^7$. While for the MEI model, we have:

$$\frac{d\theta_d}{d\theta} = \frac{(1+3k_1\chi^2+5k_2\chi^4)}{(1+r_d^2)}\frac{d\chi}{d\theta}, \tag{13}$$

$$\frac{d^2\theta_d}{d\theta^2} = \frac{\mathcal{U}-\mathcal{V}}{(1+r_d^2)^2}, \tag{14}$$

$$\mathcal{V} = 2r_d(1+3k_1\chi+5k_2\chi^4)^2(\frac{d\chi}{d\theta})^2, \tag{15}$$

$$\mathcal{U} = ((6k_1\chi+20k_2\chi^3)(\frac{d\chi}{d\theta})^2 + (1+3k_1\chi^2+5k_2\chi^4)\frac{d^2\chi}{d\theta^2})(1+r_d^2), \tag{16}$$

where $\frac{d\chi}{d\theta} = \frac{1+\xi\cos\theta}{(\cos\theta+\xi)^2}$ and $\frac{d^2\chi}{d\theta^2} = \frac{\sin\theta(2+\xi\cos\theta-\xi^2)}{(\cos\theta+\xi)^3}$. Here, $d^n\theta_d/d\theta^n (n>2)$ also have analytical expressions for both the Kannala-Brandt and MEI models. When the linear approximation is used, $k_\theta = d\theta_d/d\theta$ so the computation is considerably simplified.

Update Scaling Factors and Rotations. Generally, random variables that conform to a Gaussian distribution can retain this property even after applying affine transformations. Since the stretching operation is an affine transformation in our method, the covariance matrix of the resultant Gaussian distribution is defined as:

$$\boldsymbol{\Sigma}' = \boldsymbol{S}_\theta \boldsymbol{S}_\phi \boldsymbol{\Sigma} \boldsymbol{S}_\phi^T \boldsymbol{S}_\theta^T, \tag{17}$$

where $\boldsymbol{S}_\theta$ and $\boldsymbol{S}_\phi$ are the stretching matrices in the polar and tangential directions. Besides, the three scaling factors and the quaternion of the new 3D Gaussian are defined as:

$$s_i = \sqrt{\lambda_i}, \quad i \in \{1,2,3\}, \tag{18}$$

$$\boldsymbol{q} = quat\left([\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_1} \times \boldsymbol{v_2}]\right), \tag{19}$$

where λ_i is the eigenvalue of Σ', $\boldsymbol{v}_i$ is the eigenvector of Σ', and $\boldsymbol{v_3}$ is replaced by $\boldsymbol{v_1} \times \boldsymbol{v_2}$ to ensure that these vectors form a right-hand system.

3.3. UniGaussian for Driving Scene Reconstruction

Fig. 2 depicts an overview of our approach to learning unified Gaussian representations from multiple camera models for driving scene reconstruction.

Driving Scene Gaussians. With input images from different cameras, we represent a driving scene with composite 3D Gaussians[1]. Following [32, 36, 37], we decompose the scene into background Gaussians for static scenes, dynamic Gaussians for moving objects and sky Gaussians for distant regions. Besides, we employ LiDAR point clouds to initialize 3D Gaussians by accumulating all LiDAR frames and projecting points onto images for color extraction. This provides a better representation of the scene geometry and enables the optional LiDAR simulation in our framework.

Modeling Multiple Sensors and Modalities. To account for the distortion of fisheye images, we applied a series of affine transformations to further process 3D Gaussians for fisheye cameras as introduced in Sec. 3.2 while pinhole cameras skip these transformations. These processed 3D Gaussians are rendered with a unified tile-based rasterizer.

[1] In this work, "unified Gaussian representation" refers to learning a unified model for multiple sensors, while "composite Gaussians" refers to scene decomposition in the same model.

Then, to resolve the exposure difference between pinhole and fisheye cameras, we applied camera-dependent scaling and biases factors to the rendering images as [36]. This models appearance difference between pinhole and fisheye cameras. Besides, we also render other modalities, including depth maps, semantic maps, and normal maps. To render these maps, each 3D Gaussian is added with the corresponding 3D logits, and then 2D maps are obtained via α-blending of these 3D logits in the rasterizer. Although LiDAR simulation may not be directly achieved by explicit 3D Gaussians, we can obtain point clouds from the rendering depth maps by extracting points based on real-world LiDAR parameters and scans. Similarly, the intensities of point clouds can be obtained by generating intensity maps from the rasterizer and extracting the corresponding points. Please refer to the supplementary material for more details of the optional LiDAR simulation.

Adaptive Density Control. Driving scene reconstruction usually requires millions of 3D Gaussians. To enhance the adaptive density control of these Gaussians, we employ the Markov Chain Monte Carlo (MCMC) sampling and relocation strategy following [12]. Specifically, we consider 3D Gaussian densification and pruning as a deterministic state transition of MCMC samples and employ a relocation strategy to dynamically adjust the position of Gaussians while preserving sample probability. With the opacity and scaling regularization, this adaptive control encourages the removal of redundant Gaussians and produces more compact scene Gaussians. Please refer to [12] for more details.

Model Optimization. We employ multiple losses to optimize our model. These losses constrain the shared 3D Gaussian representation to learn complementary information from different modalities, ensuring the consistency and complementarity of outputs across sensors and modalities. Overall, the training loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}^P_{rgb} + \mathcal{L}^F_{rgb} + \mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_n + \mathcal{L}_{reg}, \tag{20}$$

where $\mathcal{L}^P_{rgb}$ and $\mathcal{L}^F_{rgb}$ are the reconstruction losses between the ground-truth and the rendering pinhole/fisheye images following [11], $\mathcal{L}_d$ is the depth loss computed between the rendering depth and the monocular depth [9] or LiDAR depth, $\mathcal{L}_s$ is the semantic loss computed with the rendering semantic map and the predefined 2D semantic segmentation map [15], $\mathcal{L}_n$ is the normal consistency loss regularizing the rendering normal and the normal derived from the depth, and $\mathcal{L}_{reg}$ is the Gaussian opacity and scale regularization term [12] to encourage a compact Gaussian representation. Please refer to the supplementary material for more details.

4. Experiments

In our experiments, we first verify the effectiveness of the proposed rendering method in Sec. 4.1 and Sec. 4.2 and then examine the efficacy of the unified framework in Sec. 4.3.

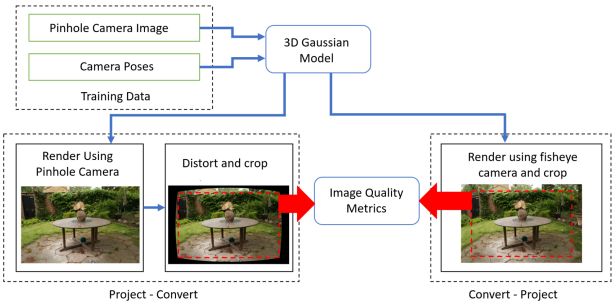


Figure 4. The flowchart for analyzing geometric errors.

		BICYCLE $FOV = 56°$ (W,H)=(4946, 3286) ROI(4391, 2917)				GARDEN $FOV = 68°$ (W,H)=(5187, 3361) ROI(4358, 2824)			
$\hat{\phi}$	$\hat{\theta}$	PSNR↑	SSIM↑	LPIPS↓	Time(ms)	PSNR↑	SSIM↑	LPIPS↓	Time(ms)
✗	✗	29.340	0.926	0.0386	122	29.805	0.902	0.0523	119
✗	1	30.752	0.936	0.0325	132	30.148	0.911	0.0455	124
1	✗	29.440	0.926	0.0381	128	29.835	0.903	0.0522	122
1	1	30.794	0.936	0.0323	149	30.189	0.911	0.0454	140
1	2	30.825	0.936	0.0322	251	30.192	0.912	0.0454	446

Table 1. Rendering geometric error analysis. $\hat{\phi}$ and $\hat{\theta}$ denote the approximation order of tangential and polar stretching ratios. "✗" means the 3D Gaussian is not stretched in that direction. "1" denotes the 1st order while "2" denotes the 2nd order.

4.1. Fisheye Rendering Geometric Error Analysis

Relative Error Analysis. Since the proposed method is a "convert-project" method, the conversion part contains some approximations. After the light rays are twisted by fisheye cameras, the distorted 3D Gaussian no longer follows an exact normal distribution. The tangential stretching ration k_ϕ does not contain any approximation, while k_θ includes the truncation error. When using the first-order approximation, the truncation error of k_θ is defined as:

$$\epsilon_{k_\theta}(\theta, \Delta\theta) = \frac{1}{2!}\frac{d^2\theta_d}{d\theta^2}\Delta\theta + \frac{1}{3!}\frac{d^3\theta_d}{d\theta^3}\Delta\theta^2 + \dots. \tag{21}$$

On the image plane, this can cause scale errors in the radial direction from the center of the image, so the relative error is defined as $\epsilon_{k_\theta}(\theta, \Delta\theta)\cos\theta_d$. However, our design is not significantly affected by this approximation. To verify this, we conduct the following experiment.

Flowchart for Analysis. We employ the method depicted in Fig. 4 to analyze the rendering geometric error introduced by these approximations. First, a 3D Gaussian model is trained using the images of a pinhole camera. Then, images of a fisheye camera are rendered in two different ways: the "convert-project" method as introduced above and the "project-convert" method that renders a pinhole image and distorts it to a fisheye image. The "project-convert" method does not have any approximation in the conversion part, so its rendering result is considered as the reference image. We use the "BICYCLE" and "GARDEN" of the Mip-NeRF360 dataset [1] in this experiment because these data

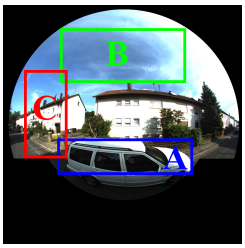
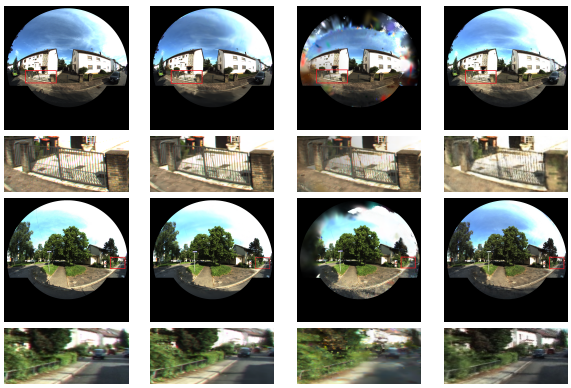


Figure 5. Image zones.

		Instant-NGP [22]	NeRFacto-big [25]	Zip-NeRF [2]	3DGS [11]+Undistort	Ours (w/o-s)	Ours (w-s)
	PSNR↑	21.247	13.751	12.604	12.635	24.574	**24.651**
	SSIM↑	0.773	0.548	0.504	0.508	0.815	**0.817**
	LPIPS↓	0.195	0.306	0.469	0.361	0.133	**0.130**
Zone-A	PSNR↑	22.339	15.495	15.366	17.010	22.784	**22.919**
	SSIM↑	0.617	0.412	0.405	0.425	0.688	**0.695**
	LPIPS↓	0.371	0.489	0.540	0.353	0.276	**0.265**
Zone-B	PSNR↑	21.210	20.917	19.228	12.437	30.830	**30.860**
	SSIM↑	0.867	0.830	0.797	0.710	0.869	**0.869**
	LPIPS↓	0.127	0.155	0.190	0.340	0.116	**0.115**
Zone-C	PSNR↑	22.401	14.768	11.978	11.637	25.720	**25.861**
	SSIM↑	0.790	0.508	0.426	0.426	0.838	**0.842**
	LPIPS↓	0.154	0.266	0.510	0.468	0.099	**0.093**
	Training time	**12m**	3h28m	37m	26m	36m	58m
	Rendering FPS	0.26	0.23	0.28	121.37	**138.89**	39.06
	3DG Num	–	–	–	697K	897K	**634K**

Table 2. Results of fisheye camera simulation on KITTI-360. "Ours(w-s)" and "Ours(w/o-s)" denote our method w/ and w/o stretching. "Undistort" denotes fisheye images are rectangularized for reconstruction and the scene is rendered with the pinhole camera model and distorted to fisheye images. For a fair comparison, all methods do not use LiDAR or multimodal optimization.



Ground truth Ours 3DGS+Undistort Instant-NGP

Figure 6. Rendering results of fisheye simulation on KITTI-360.

have enough multi-view images and accurate camera poses that enable rendering from any point of view.

Results Analysis. The results are presented in Tab. 1. From these results, we can see that the stretching of 3D Gaussians is effective for improving the quality of rendering. This proves the correctness of our rendering algorithm's derivation because the approximations do not degrade the image quality. Moreover, in our method, most of the rendering time is spent on affine transformations of 3D Gaussians, while rasterization only takes around 3 ms. Note that the time consumption shown in Tab. 1 is for images with large resolutions, but for autonomous driving scenes, 3D Gaussians are projected within a frustum and the image resolution is less than 4358×2824 so the rendering can be done in real time. Besides, although the 2nd order approximation of the polar stretching ratio gives an improvement on rendering quality, it greatly increases the rendering time.

4.2. Driving Scene Fisheye Camera Simulation

In this experiment, we verify the effectiveness of the proposed fisheye rendering method on the KITTI-360 dataset [16]. We compare the image quality of both global image and three representative local zones, namely, A, B, and C, as illustrated in Fig. 5. Details of the experimental setup can be found in the supplementary material.

Ground truth GT(local) Ours(w-s) Ours(w/o-s)

Figure 7. Rendering results of our approach w/ and w/o stretching.

We present the quantitative results in Tab. 2 and the qualitative results in Fig. 6. Overall, our approach achieves the best rendering quality both globally and locally (at zones A, B and C), while maintaining real-time rendering speed. In comparison, 3DGS+Undistort performs poorly in fisheye simulation, indicating that direct distortion is not a suitable choice. Besides, from Tab. 2 and Fig. 7, we can see that our method without stretching yields slightly worse image quality, especially for texture details, but achieves significantly faster rendering speed of 138 FPS. Also, skipping stretching increases the number of Gaussians by 40%, which increases the risk of GPU memory overflow. The reason is that small Gaussians are insensitive to changes in the distribution caused by light rays distortion and the optimizer tends to approximate the scene with a large number of small 3D Gaussians in exchange for better rendering quality.

4.3. Multiple Camera Model Simulation

In this experiment, we evaluate our UniGaussian framework for driving scene simulation from multiple camera models. We conduct our experiments on KITTI-360 [16], because

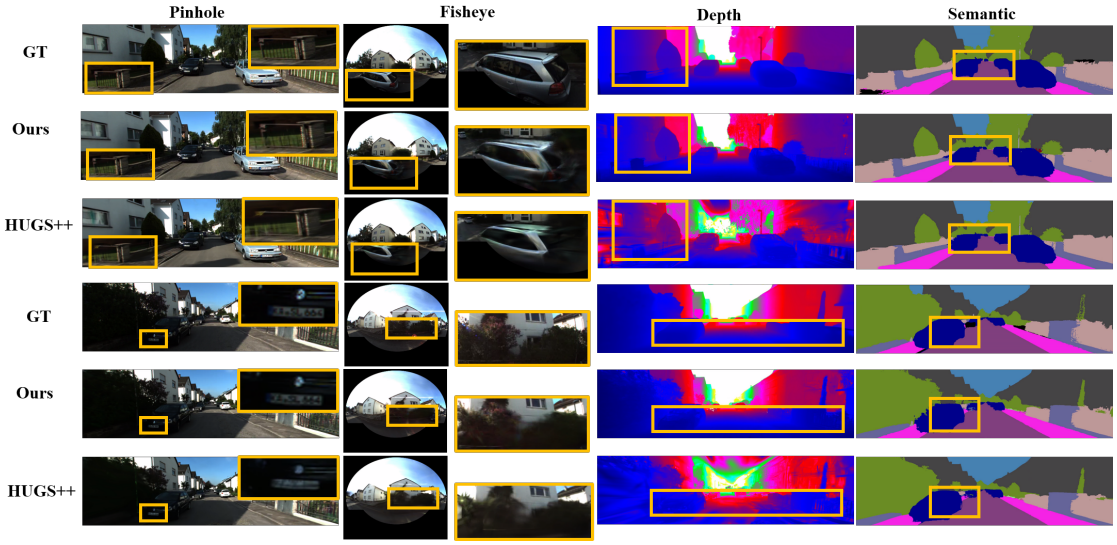


Figure 8. Rendering results of multiple camera model simulation on KITTI-360. We highlight the key differences in yellow boxes.

Method	Pinhole			Fisheye		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Instant-NGP[22]	24.6	0.808	0.181	-	-	-
UniSim-SF[26, 33]	24.9	0.812	0.184	-	-	-
AlignMiF[26]	25.3	0.826	0.164	-	-	-
Ours(pinhole)	**25.7**	**0.881**	**0.102**	-	-	-
HUGS++[36]	25.2	0.866	0.146	24.9	0.879	0.241
Ours	**26.1**	**0.886**	**0.099**	**26.2**	**0.897**	**0.185**

Table 3. Results of multiple camera model simulation on KITTI-360. "HUGS++" denotes using our method for additionally training fisheye cameras, and for a fair comparison, "HUGS++" uses the same initialization as our approach. Some results are borrowed from [26].

it is a real-world autonomous driving dataset that provides both pinhole and fisheye images, while the other commonly used datasets provide only one type of image.

As shown in Tab. 3, for pinhole camera simulation, for fair comparison with AlignMiF and UniSim-SF, we also report the results of our approach with pinhole camera only. We can see that both ours and ours(pinhole) achieve better results than the state-of-the-art driving scene reconstruction methods, such as HUGS, AlignMiF and UniSim-SF. As for fisheye camera simulation, we add our fisheye rendering method to HUGS (named HUGS++) because HUGS does not support fisheye cameras. The results in Tab. 3 show that our approach outperforms the modified method. Besides, as shown in Fig. 8, our approach generates better rendering images with more fine-grained details. For example, our approach renders the license plate, the fence, the taillight, *etc*., while HUGS++ generates worse rendering images. Moreover, we also visualize some semantic and depth maps in Fig. 8. We can see that our multimodal outputs achieve slightly better holistic 3D scene understanding.

Component	Pinhole PSNR↑	Fisheye PSNR↑
Ours	**26.10**	**26.19**
Ours w/o depth supervision	25.96	**26.19**
Ours w/o semantic supervision	26.01	25.85
Ours w/o normal supervision	26.02	26.17
Ours w/o adaptive control and $\mathcal{L}_{reg}$	25.76	25.75

Table 4. Ablation study on KITTI-360.

4.4. Ablation study

The ablation studies of the stretching and approximation operations of the proposed rendering method are discussed in Sec. 4.1 and Sec. 4.2. In this experiment, we further evaluate other components of our approach. From Tab. 4, we can see that our approach with all components performs the best and the supervisions from different modalities, including depth, semantic, and normal, are helpful for model optimization. These modalities also facilitate better holistic understanding of driving scenes as shown in Fig. 8. Besides, from the last row of Tab. 4, we can see that without the adaptive control and $\mathcal{L}_{reg}$, the PSNRs are slightly worse, indicating the importance of learning a compact representation in driving scene reconstruction.

5. Conclusion

In this work, we present a new differentiable rendering method of 3D Gaussians tailored for fisheye cameras and propose a new framework for learning unified 3D Gaussians of driving scenes from multiple camera models. Our method enables holistic driving scene understanding by modeling multiple sensors and modalities. Our experimental results on real-world autonomous driving dataset verify the effectiveness of the proposed method.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 6

[2] 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. 7, 1, 2

[3] 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. 1, 2, 3

[4] Jiho Choi, Gyutae Hwang, and Sang Jun Lee. Dico-nerf: Difference of cosine similarity for neural rendering of fisheye driving scenes. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7850–7858. IEEE, 2024. 4

[5] Fletcher Dunn and Ian Parberry. *3D math primer for graphics and game development*. CRC Press, 2011. 5

[6] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2

[7] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023. 2

[8] Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *arXiv preprint arXiv:2306.04988*, 2023. 2, 3

[9] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024. 6, 2

[10] Juho Kannala and Sami S Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1335–1340, 2006. 3

[11] 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, 6, 7, 1

[12] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *arXiv preprint arXiv:2404.09591*, 2024. 6, 2

[13] Ren Komatsu, Hiromitsu Fujii, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. 360 depth estimation from multiple fisheye images with origami crown representation of icosahedron. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10092–10099. IEEE, 2020. 2

[14] Jia Li, Jinming Su, Changqun Xia, and Yonghong Tian. Distortion-adaptive salient object detection in 360° omnidirectional images. *IEEE Journal of Selected Topics in Signal Processing*, 14(1):38–48, 2019. 2

[15] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 2, 6

[16] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 7, 1

[17] Zimu Liao, Siyan Chen, Rong Fu, Yi Wang, Zhongling Su, Hao Luo, Linning Xu, Bo Dai, Hengjie Li, Zhilin Pei, et al. Fisheye-gs: Lightweight and extensible gaussian splatting module for fisheye cameras. *arXiv preprint arXiv:2409.04751*, 2024. 2, 1

[18] Yibo Liu, Kelly Zhu, Guile Wu, Yuan Ren, Bingbing Liu, Yang Liu, and Jinjun Shan. Mv-deepsdf: Implicit modeling with multi-sweep point clouds for 3d vehicle reconstruction in autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8306–8316, 2023. 3

[19] Yibo Liu, Zheyuan Yang, Guile Wu, Yuan Ren, Kejian Lin, Bingbing Liu, Yang Liu, and Jinjun Shan. Vqa-diff: Exploiting vqa and diffusion for zero-shot image-to-3d vehicle asset generation in autonomous driving. In *European Conference on Computer Vision*, 2024. 3

[20] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950. IEEE, 2007. 3, 4

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

[22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 7, 8, 1

[23] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 2, 3

[24] Hazem Rashed, Eslam Mohamed, Ganesh Sistu, Varun Ravi Kumar, Ciaran Eising, Ahmad El-Sallab, and Senthil Yogamani. Generalized object detection on fisheye cameras for autonomous driving: Dataset, representations and baseline. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2272–2280, 2021. 2

[25] 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. 7, 1

[26] Tang Tao, Guangrun Wang, Yixing Lao, Peng Chen, Jie Liu, Liang Lin, Kaicheng Yu, and Xiaodan Liang. Alignmif: Geometry-aligned multimodal implicit field for lidar-camera joint synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21230–21240, 2024. 1, 2, 3, 8

[27] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. Bifuse: Monocular 360 depth estimation via bi-projection fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2020. 2

[28] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 2

[29] Ziyang Xie, Junge Zhang, Wenye Li, Feihu Zhang, and Li Zhang. S-nerf: Neural radiance fields for street views. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3

[30] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsegs: Real-time 360 {\deg} sparse view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00206*, 2023. 2

[31] Xu Yan, Haiming Zhang, Yingjie Cai, Jingming Guo, Weichao Qiu, Bin Gao, Kaiqiang Zhou, Yue Zhao, Huan Jin, Jiantao Gao, et al. Forging vision foundation models for autonomous driving: Challenges, methodologies, and opportunities. *arXiv preprint arXiv:2401.08045*, 2024. 2

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

[33] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023. 1, 2, 3, 8

[34] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9308–9318, 2019. 1

[35] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2

[36] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21336–21345, 2024. 1, 2, 3, 5, 6, 8

[37] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024. 2, 3, 5