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

Supplementary Material

This supplementary material is organized as follows: in Sec. 6, we present more details of the experimental setups; in Sec. 7, we provide the implementation details of our approach; in Sec. 8, we present the experimental comparison with Fisheye-GS [17]; in Sec. 9, we present results of novel view synthesis with lane shift; in Sec. 10, we discuss the details of our approach to Lidar simulation; in Sec. 11, we discuss the key difference between NeRF-based methods and our approach; in Sec. 12, we discuss the limitation and future work of our approach.

6. Experimental Setups

6.1. Experimental Setup for Fisheye Rendering Geometric Error Analysis

The pinhole and fisheye cameras are set to have the same FOV and resolution during rendering. To ensure the rendering quality of the "project-convert" method, the FOV is set close to the FOV of the training data. The resolution needs to be set to a large value because the pinhole image will shrink to the center area after distortion. The 3DGS hyper-parameters are mainly set following [11].

6.2. Experimental Setup for Driving Scene Fisheye Camera Simulation

Dataset. We conduct experiments on the KITTI-360 dataset [16]. KITTI-360 contains images captured from fisheye cameras in real-world driving scenes and provides camera calibration and ego-vehicle pose. We select a segment of 221 images (frame ids 227-447 from "2013_05_28_drive_0000_sync") and evenly select every eighth image as the test set while the others are used as the training set. In addition to comparing the global image quality, three local zones (A, B, C) are compared. As shown in Fig. 5, zone "A" (Blue) is the bottom area where objects appear usually very close to the camera, zone "B" (Green) is the sky with little texture where the geometry is not very accurate, and zone "C" (Red) is far away from the camera with the large distortion in this region.

Camera Model Conversion. The MEI camera model is the only model provided for the fisheye camera on KITTI-360, so to evaluate our method on both the MEI and Kannala-Brandt models, we convert the MEI model to the Kannala-Brandt model. Specifically, the conversion is an approximation of $\theta_d(\theta)$ determined by Eq. (3) of the main paper with the series expansion Eq. (2) of the main paper. Note that when $\theta = 0$, the first derivative of Eq. (2) equals 1, while the first derivative of Eq. (3) is $1/(1+\xi)$. To eliminate this difference, Eq. (3) is modified as:

$$\theta_d = \arctan r_d = \arctan\left(\left(\chi + k_1\chi^3 + k_2\chi^5\right)(1+\xi)\right). \quad (22)$$

Correspondingly, the focal length of Kannala-Brandt is set as $f_x = \gamma_1/(1+\xi)$ and $f_y = \gamma_2/(1+\xi)$. The first step of the conversion is to generate a set of θ and θ_d by using Eq. (22), where $\theta \in (0, \pi/2]$. Then, k_i(i=1,2,3,4) of the Kannala-Brandt model can be estimated by the least squares method. In our experiments, we test both the MEI and Kannala-Brandt camera models but the results are almost the same, so we by default report the results with the MEI model.

Compared Methods. We compare our approach with three NeRF-based methods, including Instant-NGP [22], Nerfacto-big [25], Zip-NeRF [2], and one 3DGS-based method, namely "3DGS [11]+Undistort". To adapt 3DGS for driving scene reconstruction with fisheye images, we rectangularize fisheye images for reconstruction and render scene with the pinhole camera model and distort to fisheye images (named 3DGS+Undistort). Besides, the NeRF-based methods are implemented based on [25].

6.3. Experimental Setup for Multiple Camera Model Simulation

Dataset. We conduct our experiments on KITTI-360 [16], because 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. For evaluation, we select four sequences as [26]. Each sequence contains 64 frames, and we select every fourth frame as the test set while the others are used as the training set. Note that although there are fisheye datasets for autonomous driving, such as Woodscape [34], they are constructed for perception tasks (*e.g.*, object detection, segmentation, *etc.*) rather than driving scene reconstruction, so they often only have discrete frames and lack continuous video sequence data. In contrast, KITTI-360 provides a complete real-world autonomous driving sensor suite, including fisheye, pinhole and LiDAR, so evaluation on KITTI-360 for multiple camera model simulation is comprehensive to verify the effectiveness of our approach.

Compared Methods. To the best of our knowledge, no existing driving scene reconstruction method simulates both pinhole and fisheye cameras in a unified framework.

Methods	Alameda	Berlin	London	Nyc	Average
Ours	**26.0**	23.8	**27.9**	**22.3**	**25.0**
Fisheye-GS [17]	24.1	**24.2**	25.1	20.3	23.4

Table 5. Comparison with Fisheye-GS on the Zip-NeRF(fisheye) dataset. Results are in terms of PSNR↑.

HUGS [36] is a state-of-the-art 3DGS-based method for driving scene reconstruction and 3D scene understanding. We therefore use it as a baseline method for comparison. We modify HUGS based on the official code and the paper and additionally use our differentiable rendering method to train HUGS for fisheye camera simulation. For the other compared driving scene simulation methods, we borrow the results from [26]. Note that although there are some other driving scene simulation methods, they are mostly designed for pinhole camera simulation, so we choose the representative HUGS [36], AlignMiF[26] and UniSim-SF[26, 33] for comparison in our experiments. Modifying existing state-of-the-art methods for fisheye camera simulation in autonomous driving is beyond the scope of this work.

7. Implementation Details

We implement our approach using python and pytorch. Following [11, 36], we set the initial position learning rate to 1.6×10^{-4} with a decay to 1.6×10^{-6} at the last iteration. Both scaling and rotation learning rates are set to 0.001, the learning rate of the spherical harmonics feature is set to 0.0025 and the opacity learning rate is set to 0.05. Adaptive Density Control is performed every 100 iterations starting from 500 iterations. For composite scene Gaussians, the background Gaussians are initialized with LiDAR point clouds, the dynamic Gaussians are randomly initialized, and the sky Gaussians are uniformly initialized in a distant (more than 100 meters) spherical region for simplicity, but more advanced sky cubemaps [32] can be used. The training loss $\mathcal{L}$ of our approach is defined in Eq. (20) of the main paper. The details of these losses and regularization terms are listed below.

Image Losses $\mathcal{L}^P_{rgb}$ and $\mathcal{L}^F_{rgb}$. They are the reconstruction losses between the ground-truth and the rendering pinhole/fisheye images, which are defined as:

$$\mathcal{L}_{rgb} = (1 - \lambda_{rgb})\mathcal{L}_1 + \lambda_{rgb}\mathcal{L}_{SSIM}, \tag{23}$$

where $\mathcal{L}_1$ is the L1 loss, $\mathcal{L}_{SSIM}$ is a D-SSIM term [11], and λ_{rgb} is set to 0.2 following [11].

Depth Loss $\mathcal{L}_d$. It is the depth loss computed between the rendering depth and the monocular depth $\mathcal{L}_{d-mono}$ or LiDAR depth $\mathcal{L}_{d-lidar}$. Although the depth derived from LiDAR is accurate, it can only supervise the masked region with the projected point clouds. On the other hand, the monocular depth is coarse but can supervise the whole depth map. Thus, we define $\mathcal{L}_d$ as:

$$\mathcal{L}_d = \mathcal{L}_{d-lidar} + \mathcal{L}_{d-mono}, \tag{24}$$

where $\mathcal{L}_{d-lidar}$ is the L1 loss between the depth derived from LiDAR point clouds and the masked region from the rendered depth map, and $\mathcal{L}_{d-mono}$ is the Pearson depth loss [30] between the rendered depth map and the monocular depth map computed with [9].

Semantic Loss $\mathcal{L}_s$. It is the semantic loss for the rendering semantic map M_s and the predefined 2D semantic segmentation map [15]. This is implemented with the cross-entropy loss to classify the semantic logits and its weight is set to 0.01. This loss helps to generate semantic maps for driving scenes, improving the holistic driving scene understanding.

Normal Loss $\mathcal{L}_n$ It is the normal consistency loss regularizing the rendering normal N_p and the normal derived from the depth N_d. It encourages a better geometric representation of the driving scene. We define it as:

$$\mathcal{L}_n = \mathcal{F}_m(\|1 - N_p^T N_d\|_1), \tag{25}$$

where $\mathcal{F}_m$ denotes the mean operation.

Opacity and Scale Regularization Term $\mathcal{L}_{reg}$. Following [12], we employ the Gaussian opacity and scale regularization term to encourage a compact scene Gaussian representation. It is defined as:

$$\mathcal{L}_{reg} = \lambda_{reg}\left(\mathcal{F}_m(|o|) + \mathcal{F}_m(|s|)\right), \tag{26}$$

where o are the Gaussian opacities, s are the Gaussian scales and λ_{reg} is set to 0.01.

8. Experimental Comparison with Fisheye-GS

As discussed in Sec. 2, Fisheye-GS [17] is only based on ideal camera models with equidistant projection, which hinders its use in driving scene reconstruction because fisheye cameras in driving scene are usually generic models and have large FOVs. Since Fisheye-GS is a state-of-the-art method adapting 3DGS to fisheye cameras, it would be interesting to compare our differentiable fisheye rendering method with Fisheye-GS. To this end, we conduct experimental comparison on Zip-NeRF(fisheye) [2] with four

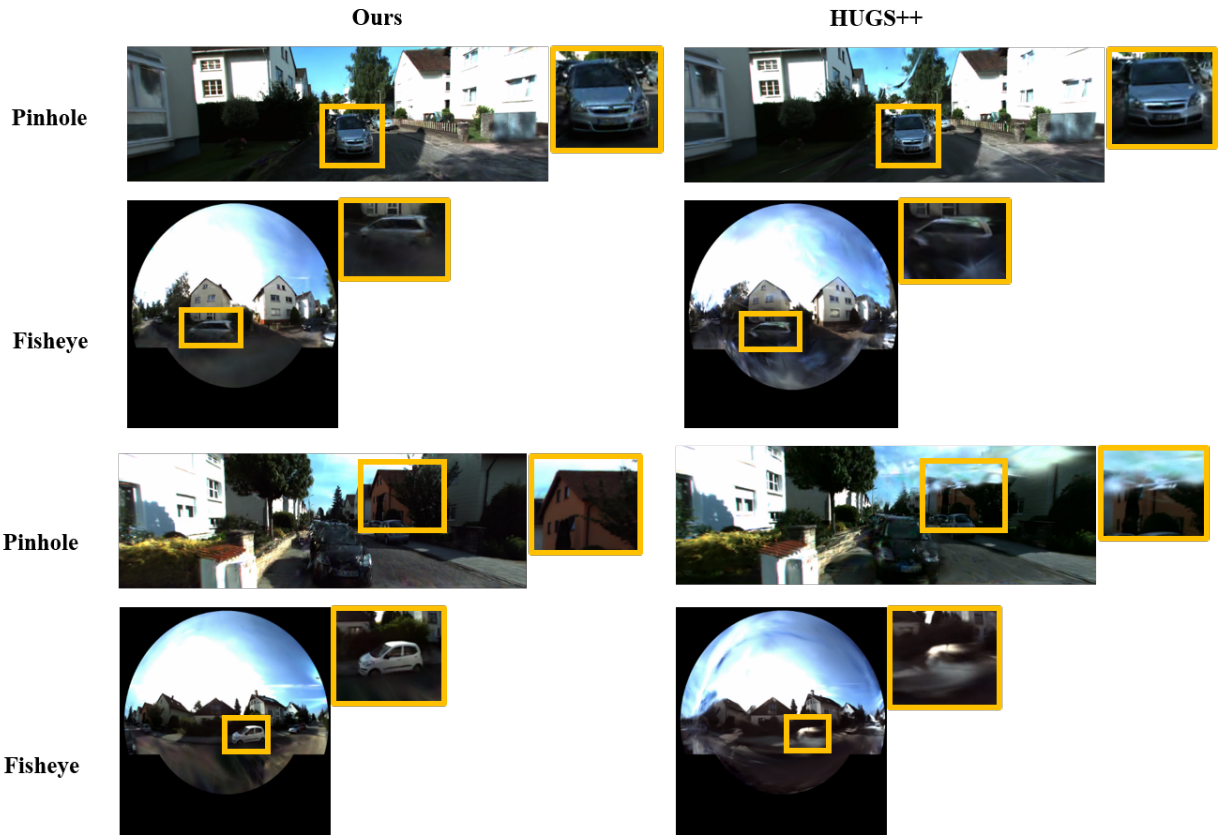

Figure 9. Qualitative results of novel view synthesis with lane shift on KITTI-360.

Methods	Pinhole FID↓@3m	Fisheye FID↓@3m
Ours	182.7	193.7
HUGS++	191.7	300.8

Table 6. Quantitative results of novel view synthesis with lane shift on KITTI-360.

large scenes, namely, Alameda, Berlin, London, and Nyc. These scenes are respectively captured with a fisheye lens of 180 degree. We select every eighth frame as the test set while the others are used as the training set. As shown in Tab. 5, our approach achieves better performance compared with Fisheye-GS. Specifically, in the Alameda, London and Nyc scenes, our approach achieves significantly better results, while in the Berlin scene, our approach achieves comparable performance compared with Fisheye-GS. On average, our approach yields PSNR of 25.0 dB while Fisheye-GS obtains PSNR of 23.4 dB. These results verify the superiority of our approach over Fisheye-GS.

9. Novel View Synthesis with Lane Shift

In driving scene reconstruction, it is important to synthesize novel views after the lane shift (left or right) of the ego vehicle for real-world simulators. In this experiment, we further present the results of novel view synthesis with lane shift @ 3 meters of the ego vehicle on KITTI-360. From Tab. 6 and Fig. 9, we can see that our approach is able to synthesize high-quality rendered images for novel view synthesis with lane shift while HUGS++ generates significantly worse results. For example, the FID of our approach is significantly better than HUGS++ and the buildings and vehicles in the rendered images of our approach are much clearer and more complete.

10. Details of the Optional LiDAR Simulation

In this section, we discuss the potential of our approach to LiDAR point clouds simulation. As mentioned in the main paper, our unified framework can be extended to simulate point clouds by extracting points from the rendering depth maps. This is achieved by predefining the LiDAR scans based on real-world LiDAR parameters and mapping the points of the scans from the world coordinate space to the camera coordinate space. By default, depth maps are associated with the pinhole camera coordinate space, but KITTI-360 only has two front pinhole cameras with narrow FOVs so the rendering depth maps cannot cover all point clouds. To solve this problem, we further render depth maps

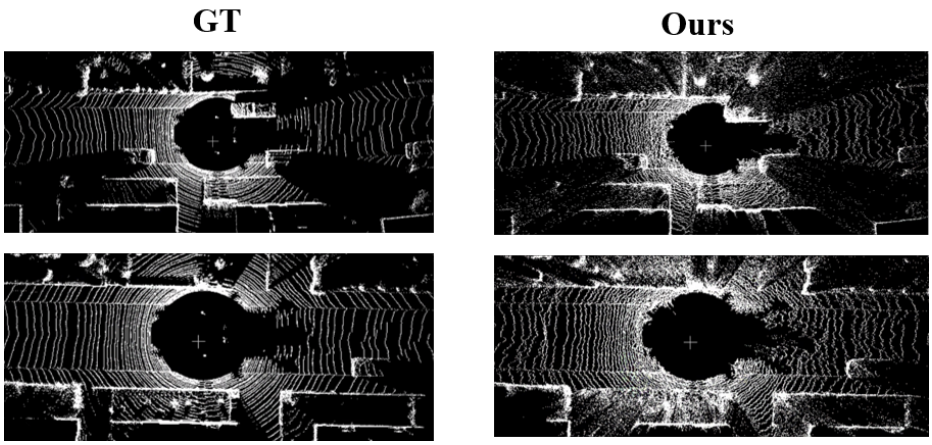

Figure 10. Visualization of simulated LiDAR point clouds on KITTI-360.

from eight pseudo cameras, covering the left, right, front and back directions, as well as downward towards the road from each of these directions. In this way, we extract each point cloud from the nearest depth map. Furthermore, to simulate the intensities of point clouds, we render intensity maps via α-blending of the 3D logits of 3D Gaussians in the rasterizer. Then, the intensities of point clouds are extracted at the corresponding locations on the intensity maps. To facilitate model optimization for LiDAR simulation, a LiDAR loss $\mathcal{L}_l$ is added to Eq. (20) of the main paper, which is defined as:

$$\mathcal{L}_l = \lambda_l \left(\mathcal{F}_m(|x_{sim} - x_{gt}|) + \mathcal{F}_m(|I_{sim} - I_{gt}|) \right), \quad (27)$$

where x_{sim} and I_{sim} are the simulated point cloud positions and intensities, x_{gt} and I_{gt} are ground truths, and λ_l is set to 0.1. In Fig. 10, we visualize some simulated LiDAR point cloud maps and intensity maps. From these results, we can observe that the simulated point cloud and intensity maps are close to the real LiDAR, especially for the nearby regions. On the other hand, we can also observe some noisy points because the indirect simulation strategy relies heavily on the accuracy of the depth maps. Note that the LiDAR simulation is not the focus of this work and more effort should be made in order to improve LiDAR simulation with 3DGS and to simulate more complex LiDAR phenomena.

11. Key Difference between NeRF-based methods and our approach

Although there have been some NeRF-based methods for fisheye rendering, such as [4], they are mostly time-consuming and cannot be combined with Gaussian splatting for driving scene simulation. Besides, in [4], Choi *et al*. show that the performance of their approach is close to Instant-NGP and NeRFacto, while our experiments in Tab. 2 show that our approach significantly outperforms Instant-NGP and NeRFacto. More importantly, we did not thoroughly compare these NeRF-based methods because the novelty of our approach is a new fisheye method to solve limitation of 3DGS and a unified framework for driving scene reconstruction, while the performance of NeRF-based methods does not diminish the novelty of our approach.

12. Limitation and Future Work

Due to the nature of explicit 3D Gaussian representations, 3DGS may not provide sufficient details when the viewing distance is close to the observation areas. This issue may be exacerbated when adapting 3DGS to fisheye cameras in driving scene reconstruction due to the capability of fisheye cameras to observe nearby vehicles and buildings along the road. However, our experiments in Sec. 4.2 show that even for zone "A" (Blue) in Fig. 5, *i.e*., the bottom area where objects appear usually very close to the camera, our approach achieves better reconstruction and renders better results compared with existing methods. To further address this limitation, future endeavors can focus on dynamically adjusting the variable lower bound of scale for 3D Gaussians observed at the close proximity based on the anticipated nearest observation distance. Such an approach would enhance the adaptability and efficiency of the rendering process, capturing nearby details. Besides, our future work also aims to develop a real-world autonomous driving simulator.