GaussianPro: 3D Gaussian Splatting with Progressive Propagation

Kai Cheng [*1] **Xiaoxiao Long** [*2] **Kaizhi Yang** [1] **Yao Yao** [3] **Wei Yin** [4] **Yuexin Ma** [5]
Wenping Wang [6] **Xuejin Chen** [1]

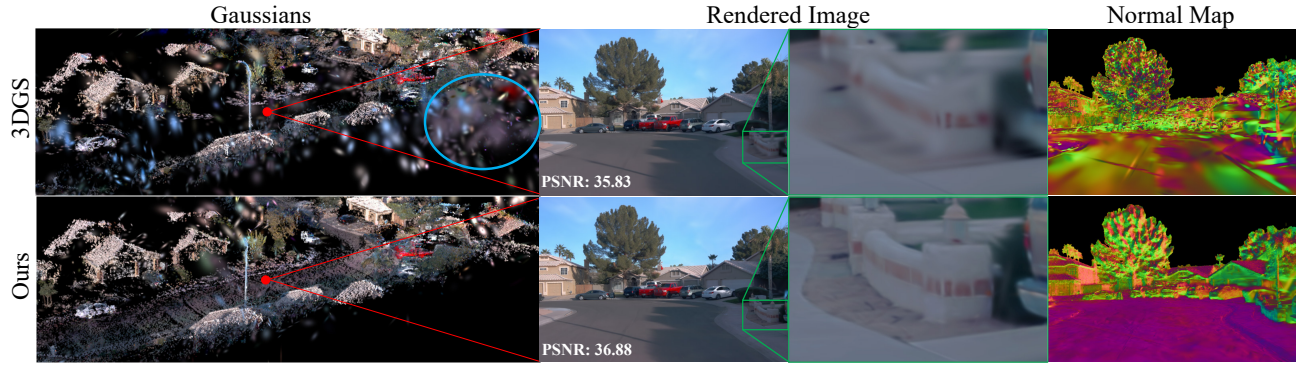


Figure 1: The sparse SfM points and less-constrained densification strategies of 3DGS pose challenges in optimizing 3D Gaussians, particularly for textureless areas. 3DGS generates inaccurate Gaussians (blue circle) that overfit the training images, leading to a noticeable performance drop in novel view rendering with erroneous geometries.

Abstract

3D Gaussian Splatting (3DGS) has recently revolutionized the field of neural rendering with its high fidelity and efficiency. However, 3DGS heavily depends on the initialized point cloud produced by Structure-from-Motion (SfM) techniques. When tackling large-scale scenes that unavoidably contain texture-less surfaces, SfM techniques fail to produce enough points in these surfaces and cannot provide good initialization for 3DGS. As a result, 3DGS suffers from difficult optimization and low-quality renderings. In this paper, inspired by classic multi-view stereo (MVS) techniques, we propose GaussianPro, a novel method that applies a progressive propagation strategy to guide the densification of the 3D Gaussians. Compared to the simple split and clone strategies used in 3DGS, our method leverages the priors of the existing reconstructed geometries of the scene and utilizes patch matching to produce new Gaussians with accurate positions and orientations. Experiments on both large-scale and small-scale scenes validate the effectiveness of our method. Our method significantly surpasses 3DGS on the Waymo dataset, exhibiting an improvement of 1.15dB in terms of PSNR. Codes and data are available at https://github.com/kcheng1021/GaussianPro.

1. Introduction

Novel view synthesis is an important but challenging task in computer vision and computer graphics. As it can generate images of novel viewpoints in the captured scene, it has extensive applications in various domains, including virtual reality (Deng et al., 2022b), autonomous driving (Yang et al., 2023a; Cheng et al., 2023), and 3D content generation (Poole et al., 2022; Tang et al., 2023). Recently, the neural radiance field (NeRF) technique (Mildenhall et al., 2020) has significantly boosted this task, achieving high-fidelity renderings without explicitly modeling 3D scenes, texture and illumination. However, due to the heavy manner of volume rendering, NeRFs still suffer from slow rendering speed, although various efforts have been made (Müller et al., 2022; Barron et al., 2022; 2023; Chen et al., 2022; Xu et al., 2022).

[*]Equal contribution [1]MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China [2]The University of Hong Kong [3]Nanjing University [4]The University of Adelaide [5]ShanghaiTech University [6]Texas A&M University. Correspondence to: Xuejin Chen <xjchen99@ustc.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has been developed to achieve real-time neural rendering. It models the scenes explicitly as 3D Gaussians with learnable attributes and performs rasterization of the Gaussians for rendering. The splatting strategy avoids time-consuming ray sampling and allows parallel computations, thus yielding high efficiency and fast rendering. However, 3DGS heavily relies on the sparse point clouds produced by Structure-from-Motion (SfM) techniques to initialize the Gaussians, e.g., their positions, colors, and shapes. Moreover, the clone and split strategies generate more Gaussians to achieve full coverage of the scene. However, the densification strategies lead to two main limitations. 1) **3DGS is sensitive to Gaussian initialization**. The SfM techniques always fail to produce 3D points and leave empty in textureless regions, and therefore the densification strategy struggles to generate reliable Gaussians to cover the scene with a poor initialization. 2) **The less-constrained densification leads to difficulties in the optimization of 3D Gaussians**, e.g., noisy geometries, and insufficient Gaussians in texture-less regions, finally degrading the rendering quality. As Figure 1 shows, the results of 3DGS contain many noisy Gaussians with wrong positions and orientations and some regions are not covered by enough Gaussians.

In this paper, we propose a novel progressive propagation strategy to produce more compact and accurate 3D Gaussians and therefore improve the rendering quality, especially in texture-less surfaces. Our key idea is to leverage the reconstructed scene geometries as priors and utilize the classic patch-matching strategy to progressively produce new Gaussians with accurate positions and orientations.

Specifically, we consider Gaussian densification in both 3D world space and 2D image space. For each input image, we render the depth and normal map by accumulating the positions and orientations of 3D Gaussians via alpha blending. Based on the observation that the neighboring pixels are likely to share similar depth and normal values, for a pixel, we iteratively propagate the depth and normal values of its neighboring pixels to this pixel to formulate a set of candidate values. We apply patch matching to pick up the best candidate value that satisfies the multi-view photometric consistency constraint, thus yielding new depth and normal for each pixel (named as propagated depth/normal). We select the pixels whose propagated depth is significantly different from the rendered depth since large differences imply that the existing 3D Gaussians may not accurately capture the true geometry. As a result, we explicitly back-project the selected pixels using the propagated depths into 3D space and initialize them as new Gaussians. Additionally, we leverage the propagated normals to regularize the orientations of 3D Gaussians, further improving the reconstructed 3D geometry and rendering quality.

By transferring accurate geometric information from well-modeled regions to under-modeled regions, our progressive propagation strategy could produce more compact and accurate 3D Gaussians and achieve better coverage of the 3D scene, as Figure 1 shows. Experiments on public datasets such as Waymo and MipNeRF360 validate that our proposed strategy significantly boosts the performance of 3DGS. In summary, the contributions of this paper include:

- We propose a novel progressive propagation strategy that guides the Gaussian densification to produce more compact and accurate Gaussians, particularly in low-texture regions.
- We additionally leverage a planar loss that provides a further constraint in the optimization of Gaussians.
- Our method achieves new state-of-the-art rendering performance on the Waymo and MipNeRF360 datasets. Our method also presents robustness to various numbers of input images.

2. Related Work

2.1. Multi-view Stereo

Multi-view stereo (MVS) aims to reconstruct a 3D model from a collection of posed images, which can be further combined with rendering algorithms to generate images under novel views. Traditional MVS methods (Campbell et al., 2008; Furukawa & Ponce, 2009; Bleyer et al., 2011; Furukawa et al., 2015; Schönberger et al., 2016; Xu & Tao, 2019) explicitly establish pixel correspondences between images based on hand-crafted image features and then optimize the 3D structure to achieve the best photometric consistency among images. Learning-based MVS methods (Yao et al., 2018; Murez et al., 2020; Long et al., 2020; Chen et al., 2019; Long et al., 2021; Ma et al., 2022; Long et al., 2022; Feng et al., 2023) implicitly build multi-view correspondences with learnable features and regress depth or 3D volume based on the features in end-to-end frameworks. In this paper, we draw inspiration from depth optimization in MVS to improve the geometry of the 3D Gaussians, thereby achieving better rendering results.

2.2. Neural Radiance Field

NeRF combines deep learning techniques with the 3D volumetric representation, transforming a 3D scene into a learnable continuous density field. Utilizing ray marching in volume rendering, NeRF is able to achieve high-quality novel view synthesis without explicit modeling of the 3D scene and illumination. To further improve the rendering quality, some approaches (Barron et al., 2021; Xu et al., 2022; Barron et al., 2023) directly improve the point sampling strategy in ray marching for more accurate model-

ing of the volume rendering process. Others (Barron et al., 2022; Wang et al., 2023) improve rendering by reparameterizing the scene to generate a more compact scene representation and lead to an easier learning process. Additionally, regularization terms (Deng et al., 2022a; Yu et al., 2022) could be introduced to constrain the scene representation towards a closer approximation of real geometry. Despite these advancements, NeRF still incurs high computational costs during rendering. Since NeRF employs MLPs to represent the scene, the computation and optimization of any point in the scene are dependent on the entire MLP. Many works propose novel scene representations to accelerate rendering. They replace MLPs with sparse voxels (Liu et al., 2020; Fridovich-Keil et al., 2022), hash tables (Müller et al., 2022), or triplane (Chen et al., 2022), allowing the computation and optimization of each point to be localized to the corresponding local region of the scene. Although these methods significantly improve rendering speed, real-time rendering is still challenging due to the inherent ray marching strategy in volume rendering.

2.3. 3D Gaussian Splatting

3DGS employs a splatting-based rasterization (Zwicker et al., 2002) approach to project anisotropic 3D Gaussians to 2D. It computes pixel colors by performing depth sorting and α-blending on the projected 2D Gaussians, avoiding the sophisticated sampling strategy of ray marching and achieving real-time rendering. Some concurrent works have made improvements to 3DGS. Firstly, 3DGS is sensitive to sampling frequency, i.e., changing the camera's focal length or camera distance may lead to rendering artifacts. These artifacts can be alleviated by introducing low-pass filtering (Yu et al., 2023) or multi-scale Gaussian representations (Yan et al., 2023). Additionally, some approaches extend 3DGS, which is primarily designed for static scenes, to dynamic scenes by modeling the motion of dynamic objects based on MLPs (Yang et al., 2023b), basis function (Li et al., 2023; Lin et al., 2023) like polynomial and Fourier Series, or rigid transformations (Zhou et al., 2023; Yan et al., 2024). Furthermore, 3DGS excessively grows Gaussians without explicitly constraining the scene's geometric structure, resulting in numerous redundant Gaussians and significant memory consumption. Some methods evaluate the contribution of Gaussians to rendering by their scales (Lee et al., 2023) or their visibility in views (Fan et al., 2023), forcing the removal of Gaussians with small contributions. Some others compress the storage of Gaussian attributes by quantization technique (Navaneet et al., 2023) or interpolating Gaussian attributes from structured grid features (Morgenstern et al., 2023; Lu et al., 2023).

Although these methods significantly reduce the storage cost of Gaussians, they do not explicitly constrain the geometry of the Gaussians. 3DGS could grow in locations far from the actual surfaces to fit different training views, resulting in the geometry redundancy of 3D Gaussians and rendering quality decrease for new viewpoints. In comparison, we propose a progressive propagation strategy to explicitly constrain the growth of 3D Gaussians near the surfaces, considering the planar structure priors in the scene. This enables Gaussians to better fit the scene geometry, achieving high-quality rendering and compact representation simultaneously.

3. Preliminaries

3DGS (Kerbl et al., 2023) models the 3D scene as a set of anisotropic 3D Guassians, which are further rendered to 2D images using the splatting-based rasterization technique (Zwicker et al., 2002). A 3D Gaussian G is defined as:

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}, \tag{1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{3\times 1}$ refers to its mean vector, $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$ refers to its covariance matrix. To ensure the positive semi-definite property of the covariance matrix during the optimization, it is further expressed as $\boldsymbol{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$, where the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times 3}$ is orthogonal, and the scale matrix $\mathbf{S} \in \mathbb{R}^{3\times 3}$ is diagonal.

To render an image from a given viewpoint, the color of each pixel $\mathbf{p}$ is calculated by blending N ordered Gaussians $\{G_i \mid i = 1, \cdots, N\}$ overlapping $\mathbf{p}$ as

$$\mathbf{c}(\mathbf{p}) = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{2}$$

where α_i is obtained by evaluating the 2D Gaussian projected from G_i at $\mathbf{p}$ (Yifan et al., 2019) multiplied with a learned opacity of G_i, and $\mathbf{c}_i$ is the learnable color of G_i. Gaussians that overlap $\mathbf{p}$ are sorted in ascending order of their depths under the current viewpoint. Through differentiable rendering techniques, all attributes of the Gaussians can be optimized end-to-end via training view reconstruction.

3DGS also employs a densification strategy to generate new Gaussians for more accurate geometric representations. At each training iteration, if the gradient backpropagated from the rendering loss to the current Gaussian exceeds a certain threshold, 3DGS considers that it does not satisfactorily represent the corresponding 3D region. If the Gaussian has a large covariance, it will split into two Gaussians. Conversely, if its covariance is small, it will be cloned. This strategy encourages 3DGS to increase the number of Gaussians to cover the captured scene.

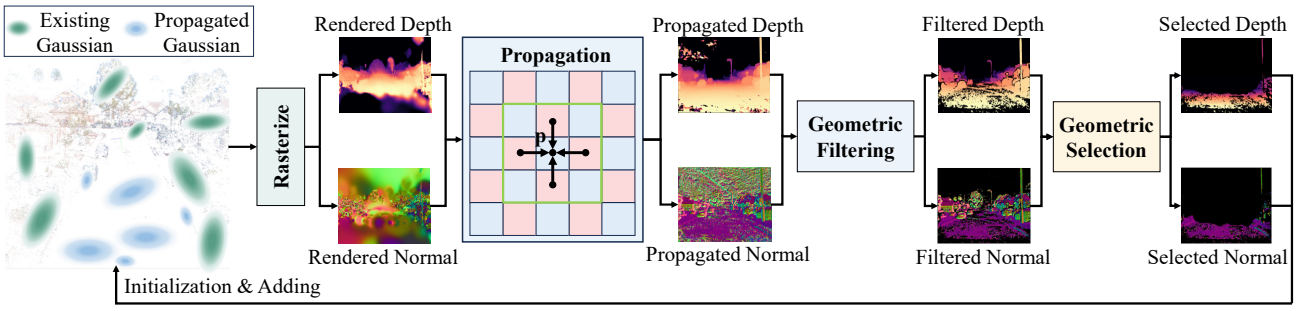


Figure 2: Progressive Propagation of Gaussian. Firstly, we render the depth and normal maps from the 3D Gaussians. Then we iteratively perform propagation operations on the rendered depths and normals to generate new depth and normal values (denoted as propagated depth and propagated normal) via patch matching techniques. We filter out the unreliable propagated depths and normals using geometric consistency, yielding filtered depths and filtered normals. Finally, we identify the regions where their rendered depths and normals significantly deviate from the filtered ones, indicating that existing Gaussians may not accurately capture the geometry and therefore need more Gaussians. Pixels in these regions are projected into the 3D space to initialize new Gaussians using the filtered depth and normal.

4. Method

In this paper, we propose a novel progressive propagation strategy to generate 3D Gaussians under the guidance of neighborhood geometry and planar structures, thereby improving rendering quality and compactness. First, instead of only working in 3D space, we propose to tackle this problem in both 3D space and 2D image space. We project 3D Gaussians to 2D to generate depth and normal maps, which are used to guide the growth of Gaussians (Sec. 4.1). Then we iteratively update the depth and normal of each pixel based on the propagated values from its neighboring pixels. Pixels whose new depth is significantly different from the initial depth are projected back to the 3D space as 3D points and these points are further initialized as new Gaussians (Sec. 4.2). Additionally, a planar loss is incorporated to further regularize the geometry of the Gaussians, yielding more accurate geometries (Sec. 4.3).

4.1. Hybrid Geometric Representation

We propose a hybrid geometric representation that combines 3D Gaussians with 2D view-dependent depth and normal maps, where the 2D representations are utilized to assist the densification of Gaussians under the guidance of the existing geometry.

Due to the irregular distribution and absence of connectivity among 3D Gaussians, it is challenging to perceive the connectivity of geometries, like searching neighboring Gaussians on a local surface. As a result, it is difficult to perceive the existing geometry to guide the Gaussian densification. Inspired by the classical MVS methods, we propose to tackle this challenge by mapping the 3D Gaussians into structured 2D image space. This mapping allows us to efficiently determine the neighbors of the Gaussians and propagate geometric information among them. Specifically, when Gaussians are located on the same local plane in 3D space, their 2D projections should also be in adjacent regions and exhibit similar geometric properties, i.e. depth and normal.

The depth value of Gaussian. For a viewpoint with camera extrinsics $[\mathbf{W}, \mathbf{t}] \in \mathbb{R}^{3\times4}$, the center $\boldsymbol{\mu}_i$ of a Gaussian G_i can be transformed into the camera coordinate system as

$$\boldsymbol{\mu}_i' = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \mathbf{W}\boldsymbol{\mu}_i + \mathbf{t}, \tag{3}$$

where z_i refers to the Gaussian depth under the current viewpoint.

The normal value of Gaussian. The covariance matrix of a Gaussian G_i is formulated as $\boldsymbol{\Sigma}_i = \mathbf{R}_i\mathbf{S}_i\mathbf{S}_i^T\mathbf{R}_i^T$. The covariance matrix $\boldsymbol{\Sigma}_i$ of a 3D Gaussian can be considered as representing an ellipsoid, where the eigenvectors of $\mathbf{R}_i$ correspond to the three axes of the ellipsoid and the scale factors of $\mathbf{S}_i$ refer to the axis lengths. According to the GaussianShader (Jiang et al., 2023), the Gaussian sphere gradually becomes flattened and approaches a plane during the optimization process. Therefore, the direction of its shortest axis can approximate the normal direction $\mathbf{n}_i$ of the Gaussian, which is induced by

$$\mathbf{n}_i = \mathbf{R}_i[r, :], r = \operatorname{argmin}\left([s_1, s_2, s_3]\right), \tag{4}$$

where $diag(s_1, s_2, s_3) = \mathbf{S}_i$, $\operatorname{argmin}(\cdot)$ is the operation to find the index of the minimum value.

Finally, the 2D depth and normal map under a viewpoint can be rendered based on α-blending defined in Eq. 2, where the attribute color $\mathbf{c}_i$ is replaced by depth z_i and normal $\mathbf{n}_i$ of each Gaussian.

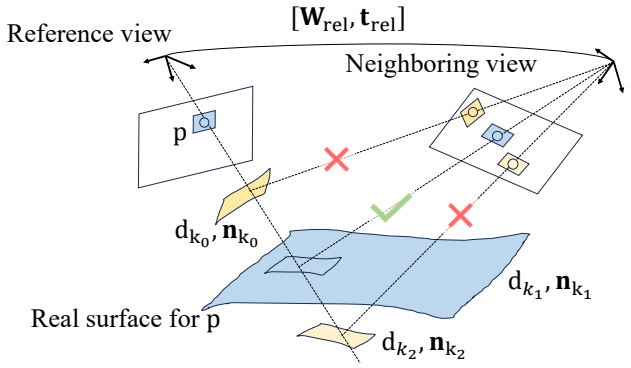


Figure 3: Patch matching. To select the best plane candidate for pixel p during propagation, we apply homography transformation on p based on each plane candidate, thus yielding the possible corresponding pixel of the neighboring view. The plane candidate that exhibits the highest color consistency between p and its possible paired pixel is chosen to be the solution. The chosen plane candidate is used to update the depth and normal of pixel p.

4.2. Progressive Gaussian Propagation

Our progressive Gaussian propagation strategy aims to propagate accurate geometry from well-modeled regions to under-modeled regions by producing new Gaussians. Figure 2 shows the pipeline. With the rendered depth maps and normal maps, we employ patch matching (Bleyer et al., 2011) to propagate the depth and normals from neighboring pixels to the current pixel, producing new depths and normals (named as propagated depth/normal). We further apply geometric filtering and selection to pick up pixels that need more Gaussians and leverage their propagated depths and normals to initialize new Gaussians.

Plane Definition. The propagation is achieved based on the patch matching technique, which requires an individual 3D plane at each pixel. Thus, the depth and normal of each pixel need to be converted to a 3D local plane first. For each pixel with its coordinate $\mathbf{p}$, the corresponding 3D local plane is parameterized as $(d, \mathbf{n})$, where $\mathbf{n}$ is the pixel's rendered normal, and d is the distance from the origin of the camera coordinate to the local plane calculated as:

$$d = z\mathbf{n}^\top \mathbf{K}^{-1}\widetilde{\mathbf{p}}, \tag{5}$$

where $\widetilde{\mathbf{p}}$ is the homogeneous coordinate of $\mathbf{p}$, z is pixel's rendered depth, and $\mathbf{K}$ refers to the camera intrinsic matrix.

Candidate Selection. After defining the 3D local plane, we select neighbors of each pixel for propagation. We follow the checkerboard pattern defined in ACMH (Xu & Tao, 2019) to select neighboring pixels. For clarity, we illustrate the propagation of a pixel with its four nearest pixels. For each pixel, a set of plane candidates $\{(d_{k_l}, \mathbf{n}_{k_l}) \mid l \in \{0, 1, 2, 3, 4\}\}$ is obtained through propagation (k_l refers to the index of pixel p and its four neighboring pixels).

Patch Matching. After obtaining the plane candidates, the optimal plane for each pixel is determined through patch matching. For a pixel p with its coordinate $\mathbf{p}$, a homography transformation $\mathbf{H}$ is performed based on each plane candidate $(d_{k_l}, \mathbf{n}_{k_l})$, which warps $\mathbf{p}$ to $\mathbf{p}'$ in the neighboring frame as:

$$\widetilde{\mathbf{p}}' \simeq \mathbf{H}\widetilde{\mathbf{p}}, \tag{6}$$

where $\widetilde{\mathbf{p}}'$ is the homogeneous coordinate of $\mathbf{p}'$, and $\mathbf{H}$ can be induced as:

$$\mathbf{H} = \mathbf{K}\left(\mathbf{W}_{\text{rel}} - \frac{\mathbf{t}_{\text{rel}}\mathbf{n}_{k_l}^\top}{d_{k_l}}\right)\mathbf{K}^{-1}, \tag{7}$$

where $[\mathbf{W}_{\text{rel}}, \mathbf{t}_{\text{rel}}]$ is the relative transformation from the reference view to the neighboring view. Finally, the color consistency of p and p' is evaluated based on NCC (Normalized Cross Correlation) (Yoo & Han, 2009). The local plane of p will be updated to the plane candidate with the best color consistency. Figure 3 also provides an intuitive visualization of this process. The propagation for plane candidates is iterated u times to transmit effective geometric information over a large region. Then the pixel's depth and normal are updated from the propagated plane, ultimately resulting in the propagated depth and normal maps in Figure 2.

Geometric Filtering and Selection. Due to the inevitable errors in the propagated results, we filter out inaccurate depth and normal through multi-view geometric consistency check (Schönberger et al., 2016) and obtain filtered depth and normal maps. Finally, we calculate the absolute relative difference between the filtered depth and rendered depth. For regions with an absolute relative difference greater than the threshold σ, we consider that existing Gaussians fail to model these regions accurately. Therefore, we project pixels in these regions back to the 3D space and initialize them as 3D Gaussians using the same initialization in 3DGS. These Gaussians are then added to the existing Gaussians for further optimization.

4.3. Plane Constraint Optimization

In the original 3DGS, the optimization only relies on image reconstruction loss without incorporating any geometric constraints. As a result, the optimized Gaussian shapes may deviate significantly from the actual surface geometry. This deviation leads to a decline in the rendering quality when viewed from a new viewpoint, particularly for large-scale scenes with limited views. As shown in Figure 4, the shape of Gaussians in 3DGS differs significantly from the road surface, resulting in severe rendering artifacts when

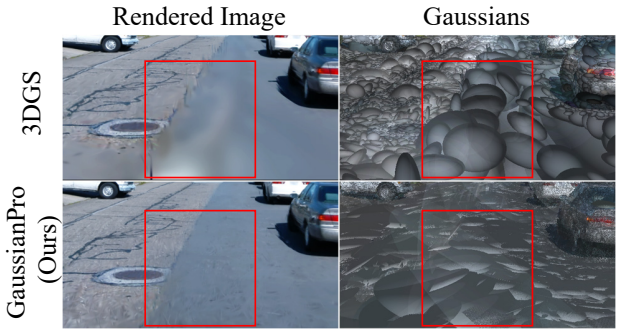


Figure 4: Visual comparisons with 3DGS on novel view synthesis. The rendered image of 3DGS contains severe artifacts since the Gaussian spheres are out of order and do not accurately model the true geometry. On the contrary, our method faithfully captures the details of the road, and its Gaussian spheres are more compact and orderly.

viewed from a novel viewpoint. In this section, we propose a planar constraint that encourages the shape of Gaussians to closely resemble the real surface.

Specifically, the propagated 2D normal map in Section 4.2 represents the orientation of the planes in the scene. We explicitly enforce the consistency between rendered normal and the propagated normal with $\mathcal{L}_1$ and angular loss as $\mathcal{L}_{\text{normal}}$:

$$\mathcal{L}_{\text{normal}} = \sum_{\mathbf{p}\in\mathcal{Q}} \|\hat{N}(\mathbf{p}) - \bar{N}(\mathbf{p})\|_1 + \left\|1 - \hat{N}(\mathbf{p})^\top \bar{N}(\mathbf{p})\right\|_1, \tag{8}$$

where $\hat{N}$ is the rendered normal map, $\bar{N}$ is the propagated normal map, and $\mathcal{Q}$ refers to the set of valid pixels after the geometric filtering in Section 4.2.

Additionally, to ensure that the shortest axis of the Gaussian could represent the normal direction, we incorporate a scale regularization loss $\mathcal{L}_{\text{scale}}$ in NeuSG (Chen et al., 2023). This loss constrains the minimum scale in Gaussian to be close to zero, effectively flattening the Gaussians towards a planar shape. Finally, the plane constraint is represented as the weighted sum of two losses:

$$\mathcal{L}_{\text{planar}} = \beta\mathcal{L}_{\text{normal}} + \gamma\mathcal{L}_{\text{scale}}. \tag{9}$$

4.4. Training Strategy

In summary, we incorporate the progressive Gaussian propagation strategy into 3DGS, activating it every m iteration in the optimization, where we set $m = 50$. The propagated normal maps are saved for computing the planar constraint loss. Our final training loss $\mathcal{L}$ consists of the image reconstruction loss $\mathcal{L}_1$ and $\mathcal{L}_{\text{D-SSIM}}$ in 3DGS with the proposed planar constraint loss $\mathcal{L}_{\text{planar}}$, as illustrated in Eq. 10.

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}} + \mathcal{L}_{\text{planar}}, \tag{10}$$

where the weight λ is set to 0.2, the same as 3DGS.

5. Experiment

5.1. Datasets and Implementation Details

Datasets. We conduct our experiments in a large-scale urban dataset Waymo (Sun et al., 2020), and the common NeRF benchmark Mip-NeRF360 dataset. (Caesar et al., 2020). On the Waymo dataset, we randomly select nine scenes for evaluation. To evaluate the performance of novel view synthesis, following the common settings, we select one of every eight images as testing images and the remaining ones as training data. We apply the three widely-used metrics for evaluation, $i.e.$, peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and the learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018).

Implementation Details. Our method is built upon the popular open-source 3DGS code base (Kerbl et al., 2023). In alignment with the approach described in 3DGS, our models are trained for 30,000 iterations across all scenes following 3DGS's training schedule and hyperparameters. Besides the original clone and split Gaussian densification strategies used in 3DGS, we additionally perform our proposed progressive propagation strategy every 50 training iteration where propagation is performed 3 times. The threshold σ of the absolute relative difference is set to 0.8. For the planar loss, we set $\beta = 0.001$ and $\gamma = 100$. All experiments are conducted on an RTX 3090 GPU. More implementation details can be found in the appendix.

5.2. Quantative and Qualitative Results

As shown in Table 1, we compare our method with the state-of-the-art (SOTA) methods, including Instant-NGP (Müller et al., 2022), Mip-NeRF360 (Barron et al., 2022), ZipNeRF (Barron et al., 2023), and 3DGS (Kerbl et al., 2023).

Results on Waymo. On the large-scale urban dataset Waymo, our method significantly outperforms others in all evaluation metrics. Due to the presence of textureless regions in street views, initializing point clouds in these regions becomes a challenge for SfM. Consequently, it is difficult for 3DGS to densify Gaussians that accurately represent the geometry of the scene in these regions. Otherwise, our propagation strategy accurately complements the missing geometry in the scene. Additionally, our planar constraint allows for better modeling of the scene's planes. Therefore, compared to the baseline 3DGS, our method significantly improves PSNR by 1.15 dB. The visual re-

Table 1: Quantitative comparisons on Waymo and MipNeRF360. We indicate the best and second best with bold and underlined respectively. 3DGS (Retrained) refers to the results obtained by 3DGS retrained with better SfM point clouds.

Method	FPS ↑	Waymo PSNR ↑	Waymo SSIM ↑	Waymo LPIPS ↓	MipNeRF 360 PSNR ↑	MipNeRF 360 SSIM ↑	MipNeRF 360 LPIPS ↓
Instant-NGP (Müller et al., 2022)	3	30.98	0.886	0.281	25.59	0.699	0.331
Mip-NeRF 360 (Barron et al., 2022)	0.02	30.09	0.909	0.262	27.69	0.792	0.237
Zip-NeRF (Barron et al., 2023)	0.09	<u>34.22</u>	<u>0.939</u>	<u>0.205</u>	**28.54**	**0.828**	**0.189**
3DGS (Kerbl et al., 2023)	<u>103</u>	33.53	0.938	0.226	27.21	0.815	0.214
3DGS (Retrained)	102	-	-	-	27.88	0.824	0.209
GaussianPro (Ours)	**108**	**34.68**	**0.949**	**0.191**	<u>27.92</u>	<u>0.825</u>	<u>0.208</u>

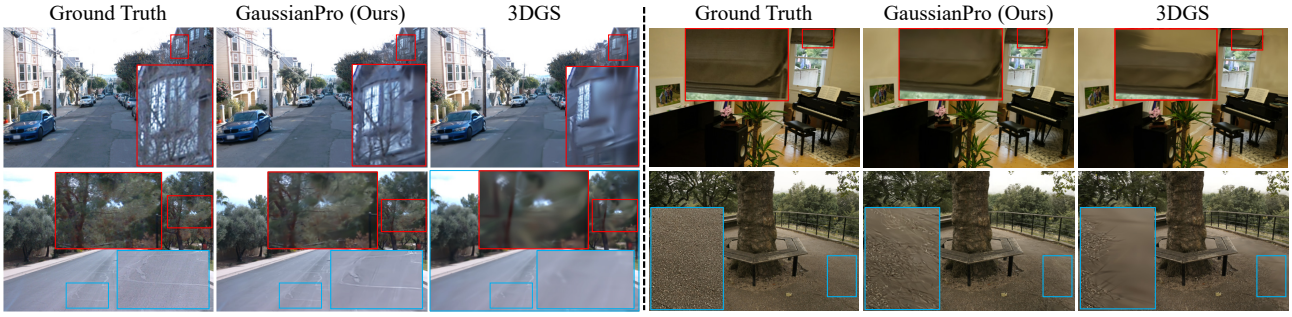


Figure 5: Rendering results on the Waymo (left) and MipNeRF360 (right) datasets. Compared to 3DGS, we have achieved a noticeable improvement in both texture-less surfaces and sharp details.

Table 2: Ablation study on the proposed propagation strategy and planar constraint.

Propagation	Planar	PSNR ↑	SSIM ↑	LPIPS ↓
✗	✗	33.53	0.938	0.226
✗	✓	34.02	0.942	0.218
✓	✗	34.48	0.946	0.203
✓	✓	**34.68**	**0.949**	**0.191**

sults presented in Figure 5 show that our method achieves sharp details and better renderings in both rich texture and texture-less regions.

Results on MipNeRF360. On the MipNeRF360, we retrain 3DGS using our generated SfM point clouds since we observed that the SfM points used in their official code base can be improved. We report the quantitative results of the original 3DGS and our retrained 3DGS in Table 1. Our method achieves comparable results with 3DGS with a slight improvement. The MipNeRF360 dataset contains quite small-scale natural and indoor scenes with rich textures, so the SfM techniques usually provide a high-quality point cloud for initialization and the simple clone and split densification strategies don't show a bottleneck in the small-scale scenes. For indoor scenes with some weak-texture surfaces, our method still shows improvement. We report results for each scene under MipNeRF360 in the appendix to further support our conclusions. As shown in the right of Figure 5, compared to 3DGS, our method achieves more accurate renderings and clear details.

5.3. Ablation Study

Effectiveness of the Propagation Strategy and Planar Constraint. We validate the effectiveness of the proposed propagation strategy and planar constraint in the Waymo dataset. As shown in Table 2, the progressive propagation strategy (the third row) brings significant improvement compared with the baseline. This improvement can be attributed to its ability to refine the geometric representation of the scene, particularly in regions where the initial 3DGS exhibits significant errors (shown in the first and second rows of Figure 6). The planar constraint can further enhance the rendering quality by accurately modeling the normals of the planes, as shown in the third row of Figure 6.

Gaussian Number. Considering that more Gaussians can better model scene details and achieve higher quality rendering, it is worth investigating whether our rendering improvements over 3DGS are primarily due to generating a larger number of Gaussians. As shown in Table 3, we present the relationship between PSNR and the number of Gaussians on the Waymo and MipNeRF360 datasets. In Waymo, we retrain 3DGS by lowering the gradient threshold for Gaussian densification as 3DGS*, resulting in the generation of a larger number of Gaussians. The rendering quality of 3DGS improves with an increased number

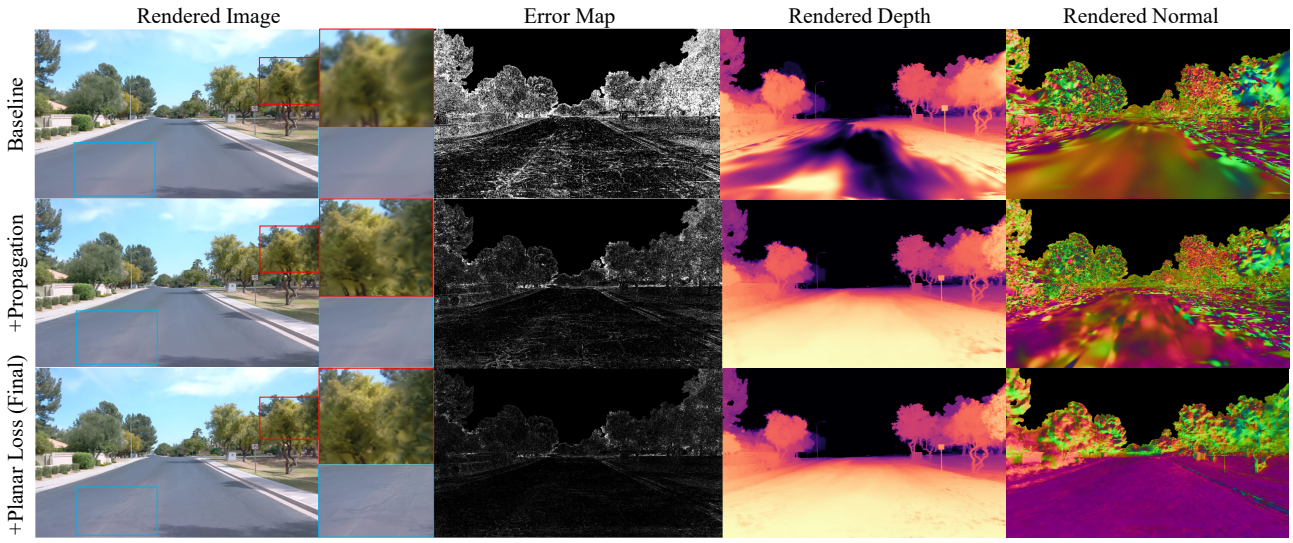


Figure 6: The progressive propagation strategy effectively enhances the geometry of the scene, resulting in improved rendering quality. The planar constraint further improves the geometry and rendering of planes.

Table 3: Ablation study on Gaussian number. 3DGS* refers to the results obtained by 3DGS retrained with a lower gradient threshold for Gaussian densification. 3DGS (Retrained) refers to the results obtained by 3DGS retrained with better SfM point clouds.

Datasets	Strategy	PSNR	Gaussians
Waymo	3DGS	33.53	**992k**
	3DGS*	33.89	1629k
	GaussianPro	**34.68**	1147k
MipNeRF360	3DGS	27.21	3362k
	3DGS (Retrained)	27.88	3009k
	GaussianPro	**27.92**	**2773k**

of Gaussians, as more Gaussians offer better fitting capabilities. However, even when the number of Gaussians in 3DGS* is larger than ours, its rendering quality remains significantly lower than ours. This highlights the importance of our strategy in densifying Gaussians with accurate positions and orientations. Furthermore, the original 3DGS utilizes fewer Gaussians than ours in the Waymo dataset since it models the street incompletely and contains many holes. In MipNeRF360, compared to the original 3DGS, the retrained 3DGS (mentioned in results on MipNeRF360 of Section 5.2) achieves better rendering quality due to more accurate SfM point clouds, even with fewer Gaussians. Compared to 3DGS, our method provides accurate geometric guidance for Gaussian densification, effectively suppressing the generation of noisy Gaussians. As a result, we achieve better rendering quality with fewer Gaussians.

Geometric Evaluation. We quantitatively demonstrate that our method improves the accuracy of the scene geometry modeled by 3D Gaussians, as shown in Table 4.

Table 4: Rendered depth evaluation of 3DGS and GaussianPro in the Waymo dataset.

Method	Abs Rel ↓	MAE(m) ↓	δ_1 ↑
3DGS	0.349	6.11	0.570
GaussianPro	**0.081**	**1.97**	**0.933**

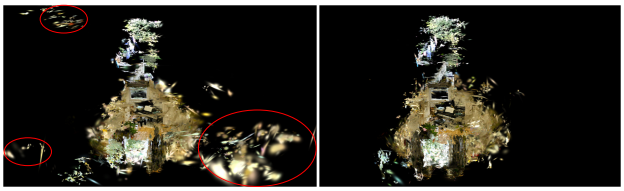
We compare the accuracy of the depth rendered by our method with that of 3DGS in the Waymo dataset. The results clearly show a significant improvement in common depth evaluation metrics (Laina et al., 2016).

The Robustness against Sparse Training Images. As the number of training images decreases, the rendering quality of neural rendering methods, including 3DGS, tends to decline. In Table 5, we present the results of training 3DGS and our method using randomly selected subsets comprising 30%, 50%, 70%, 100% of the training images from a scene in MipNeRF360. Remarkably, our method consistently achieves superior rendering results compared to 3DGS across different percentages of training images.

Efficiency Analysis. We select two typical outdoor and indoor scenes to compare the efficiency of our method with 3DGS, as shown in Table 6. We achieve a noticeable improvement in rendering quality with a slight increase in training time. In the case of the street scene, 3DGS uses large incorrect Gaussians to represent the ground, as shown in the blue circle of Figure 1, resulting in fewer Gaussians compared to our method. However, for the room scene, our method results in more compact Gaussians with less noise (also shown in Figure 7). Additionally, our method achieves a comparable real-time rendering speed as 3DGS.

Table 5: Comparison of 3DGS and ours with different training view ratios in the *room* scene of the MipNeRF360 dataset.

Method	30%			50%			70%			100%		
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
3DGS	28.45	0.896	0.216	29.97	0.912	0.203	30.87	0.921	0.194	31.71	0.919	**0.192**
GaussianPro(Ours)	**28.64**	**0.900**	**0.210**	**30.27**	**0.914**	**0.199**	**30.93**	**0.924**	**0.189**	**31.98**	**0.927**	**0.192**



3DGS GaussianPro (Ours)

Figure 7: Visualization of Gaussians in the *Room* scene of MipNeRF360 dataset. Our method contains fewer noisy Gaussians and achieves a more compact representation.

Table 6: Efficiency analysis. We compare the rendering performance and efficiency of our GaussianPro with 3DGS initialized by SfM points and MVS points.

Scene	Strategy	PSNR	Gaussians	Training	FPS
Street	SfM + 3DGS	35.05	**665k**	**40min**	**119**
	MVS + 3DGS	**36.13**	1705k	250min	75
	SfM + GaussianPro	36.08	991k	56min	108
Room	SfM + 3DGS	31.71	1537k	**59min**	105
	MVS + 3DGS	**32.05**	1832k	270min	90
	SfM + GaussianPro	31.98	**1461k**	70min	**113**

Comparison to MVS Inputs. As our method achieves better rendering quality by improving the geometry of 3D Gaussian, it raises the question of whether a similar effect can be achieved by directly inputting denser and more accurate MVS point clouds into 3DGS. To investigate this, we compare the results of optimizing 3DGS with the dense point cloud generated by the MVS method (Schönberger et al., 2016). Table 6 shows that directly inputting the MVS point cloud significantly increases the training time (approximately 4 times) due to the additional MVS process and the large number of initial Gaussians. Moreover, the number of Gaussians increases significantly, and the rendering speed noticeably decreases, despite a slight improvement in rendering quality. Contrarily, our method achieves a favorable balance between rendering quality and efficiency.

Number of Neighboring Pixels for Propagation. During progressive Gaussian propagation, we follow ACMH (Xu & Tao, 2019) to select eight neighboring pixels as candidates. We validate the impact of reducing or increasing the number of selected pixels on rendering performance, as shown in Table 7. Since the propagation of neighboring points is computed in parallel, the number of neighboring points does not affect the time consumption. A sparse selection of points cannot cover the entire neighboring areas, thus limiting the improvement in rendering quality. As the number of neighboring points increases, the rendering quality improves and finally converges.

Table 7: Ablation study on the number of neighboring pixels for propagation.

Number	PSNR ↑	SSIM ↑	LPIPS ↓	Training time
1	35.42	0.953	0.226	**56min**
2	35.55	0.954	0.226	**56min**
4	35.88	0.959	0.209	**56min**
6	36.07	0.959	0.210	**56min**
8	36.08	**0.960**	**0.204**	**56min**
10	**36.11**	**0.960**	**0.204**	**56min**

6. Conclusion

In this paper, we propose GaussianPro, a novel progressive propagation strategy to guide Gaussian densification according to the surface structure of the scene. Based on the iterative propagation process, we additionally introduce the plane constraint during optimization to encourage the Gaussians to better model planar surfaces. Our method demonstrates superior rendering results compared to 3DGS on both Waymo and MipNeRF360 datasets, while maintaining compact Gaussian representations. Our method shows significant improvements in structured scenes and remains robust to variations in the number of training images. However, our method does not specially model dynamic objects and will present artifacts on these regions like all the static Gaussian methods. In the future, the recent dynamic Gaussian techniques can be incorporated into our method as complementary components to handle dynamic objects.

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities under Grant WK3490000008.

Impact Statement

This paper presents work whose goal is to advance the field of Novel View Synthesis. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. Mip-Nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. Mip-Nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5470–5479, 2022.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. Zip-Nerf: Anti-aliased grid-based neural radiance fields. *Proceedings of the IEEE International Conference on Computer Vision*, 2023.

Bleyer, M., Rhemann, C., and Rother, C. Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*, volume 11, pp. 1–11, 2011.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. NuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Campbell, N. D., Vogiatzis, G., Hernández, C., and Cipolla, R. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I 10*, pp. 766–779. Springer, 2008.

Chen, A., Xu, Z., Geiger, A., Yu, J., and Su, H. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pp. 333–350. Springer, 2022.

Chen, H., Li, C., and Lee, G. H. NeuSG: Neural implicit surface reconstruction with 3D gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.

Chen, R., Han, S., Xu, J., and Su, H. Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1538–1547, 2019.

Cheng, K., Long, X., Yin, W., Wang, J., Wu, Z., Ma, Y., Wang, K., Chen, X., and Chen, X. UC-Nerf: Neural radiance field for under-calibrated multi-view cameras. In *The Twelfth International Conference on Learning Representations*, 2023.

Deng, K., Liu, A., Zhu, J.-Y., and Ramanan, D. Depth-supervised Nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12882–12891, 2022a.

Deng, N., He, Z., Ye, J., Duinkharjav, B., Chakravarthula, P., Yang, X., and Sun, Q. Fov-Nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864, 2022b.

Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., and Wang, Z. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

Feng, Z., Yang, L., Guo, P., and Li, B. CVRecon: Rethinking 3D geometric feature learning for neural reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17750–17760, 2023.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5501–5510, 2022.

Furukawa, Y. and Ponce, J. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

Furukawa, Y., Hernández, C., et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., and Ma, Y. GaussianShader: 3D gaussian splatting with shading functions for reflective surfaces. *arXiv preprint arXiv:2311.17977*, 2023.

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pp. 239–248. IEEE, 2016.

Lee, J. C., Rho, D., Sun, X., Ko, J. H., and Park, E. Compact 3D Gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023.

Li, Z., Chen, Z., Li, Z., and Xu, Y. Spacetime gaussian feature splatting for real-time dynamic view synthesis. *arXiv preprint arXiv:2312.16812*, 2023.

Lin, Y., Dai, Z., Zhu, S., and Yao, Y. Gaussian-flow: 4D reconstruction with dynamic 3D gaussian particle. *arXiv preprint arXiv:2312.03431*, 2023.

Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., and Theobalt, C. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

Long, X., Liu, L., Theobalt, C., and Wang, W. Occlusion-aware depth estimation with adaptive normal constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pp. 640–657. Springer, 2020.

Long, X., Liu, L., Li, W., Theobalt, C., and Wang, W. Multi-view depth estimation using epipolar spatio-temporal networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8258–8267, 2021.

Long, X., Lin, C., Wang, P., Komura, T., and Wang, W. SparseNeus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pp. 210–227. Springer, 2022.

Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., and Dai, B. Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. *arXiv preprint arXiv:2312.00109*, 2023.

Ma, Z., Teed, Z., and Deng, J. Multiview stereo with cascaded epipolar RAFT. In *European Conference on Computer Vision*, pp. 734–750. Springer, 2022.

Mildenhall, B., Srinivasan, P., Tancik, M., Barron, J., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 2020.

Morgenstern, W., Barthel, F., Hilsmann, A., and Eisert, P. Compact 3D scene representation via self-organizing gaussian grids. *arXiv preprint arXiv:2312.13299*, 2023.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022.

Murez, Z., Van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. Atlas: End-to-end 3d scene reconstruction from posed images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pp. 414–431. Springer, 2020.

Navaneet, K., Meibodi, K. P., Koohpayegani, S. A., and Pirsiavash, H. Compact3D: Compressing Gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. DreamFusion: Text-to-3D using 2D diffusion. In *The Eleventh International Conference on Learning Representations*, 2022.

Schönberger, J. L., Zheng, E., Frahm, J.-M., and Pollefeys, M. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pp. 501–518. Springer, 2016.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G. Dreamgaussian: Generative gaussian splatting for efficient 3D content creation. *arXiv preprint arXiv:2309.16653*, 2023.

Wang, P., Liu, Y., Chen, Z., Liu, L., Liu, Z., Komura, T., Theobalt, C., and Wang, W. F2-Nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4150–4159, 2023.

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., and Luo, P. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34: 12077–12090, 2021.

Xu, Q. and Tao, W. Multi-scale geometric consistency guided multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5483–5492, 2019.

Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., and Neumann, U. Point-Nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5438–5448, 2022.

Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., Lang, X., Zhou, X., and Peng, S. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024.

Yan, Z., Low, W. F., Chen, Y., and Lee, G. H. Multi-scale 3D Gaussian splatting for anti-aliased rendering. *arXiv preprint arXiv:2311.17089*, 2023.

Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.-C., Yang, A. J., and Urtasun, R. UniSim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1389–1399, 2023a.

Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., and Jin, X. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023b.

Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 767–783, 2018.

Yifan, W., Serena, F., Wu, S., Öztireli, C., and Sorkine-Hornung, O. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

Yoo, J.-C. and Han, T. H. Fast normalized cross-correlation. *Circuits, systems and signal processing*, 28: 819–843, 2009.

Yu, Z., Peng, S., Niemeyer, M., Sattler, T., and Geiger, A. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems*, 35:25018–25032, 2022.

Yu, Z., Chen, A., Huang, B., Sattler, T., and Geiger, A. Mip-Splatting: Alias-free 3D gaussian splatting. *arXiv preprint arXiv:2311.16493*, 2023.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhou, X., Lin, Z., Shan, X., Wang, Y., Sun, D., and Yang, M.-H. DrivingGaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. *arXiv preprint arXiv:2312.07920*, 2023.

Zwicker, M., Pfister, H., Van Baar, J., and Gross, M. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

A. More Implementation Details of Gaussian Progressive Propagation

Neighboring view selection. In Gaussian progressive propagation, we need to select neighboring views to evaluate color or geometric consistency and determine the propagated depth and normal. For video inputs in the Waymo dataset, we directly choose two consecutive frames before and after the current frame as neighboring views. For an unordered collection of images in MipNeRF360, we determine neighboring frames based on the number of SfM (Structure from Motion) points shared between different frames. During propagation, to improve the accuracy of depth and normals, we generate additional hypotheses for each local plane by randomly perturbing the values of distance and normal.

Sky masks. For outdoor datasets like Waymo, we use Segformer (Xie et al., 2021) to segment the sky region. Since the sky lacks precise geometric structure, we mask the sky during the propagation process to avoid Gaussian densification and plane constraint. We also masked out the sky region in the rendered depth map and normal map.

Derivation of homography transformation. The derivation of homography transformation $\mathbf{H}$ used in Sec. 4.2 is as follows. For a pixel with its coordinate $\mathbf{p}$ and its depth z, we first project it back to the 3D space as a 3D point $\mathbf{x}$:

$$\mathbf{x} = z\mathbf{K}^{-1}\widetilde{\mathbf{p}}, \tag{11}$$

where $\mathbf{K}$ is the camera intrinsic, $\widetilde{\mathbf{p}}$ is the homogeneous coordinate of $\mathbf{p}$. Based on the relative transformation $[\mathbf{W}_{\text{rel}}, \mathbf{t}_{\text{rel}}]$ between the current view and the neighboring view, we transform point $\mathbf{x}$ from the current viewpoint to the neighboring viewpoint, and then project it back to the 2D space as $\mathbf{p}'$:

$$\mathbf{p}' \simeq \mathbf{K}\left(\mathbf{W}_{\text{rel}}\,\mathbf{x} + \mathbf{t}_{\text{rel}}\right). \tag{12}$$

Since point $\mathbf{x}$ lies on a local 3D plane parameterized as $(d, \mathbf{n})$ (mentioned in Sec. 4.2), which satisfies:

$$\mathbf{n}^\top \mathbf{x} + d = 0. \tag{13}$$

Substituting Eq. 13 into Eq. 12 as:

$$\begin{aligned}
\widetilde{\mathbf{p}'} &\simeq \mathbf{K}\left(\mathbf{W}_{\text{rel}}\mathbf{x} - \frac{\mathbf{t}_{\text{rel}}\mathbf{n}^\top\mathbf{x}}{d}\right) \\
&\simeq \mathbf{K}\left(\mathbf{W}_{\text{rel}} - \frac{\mathbf{t}_{\text{rel}}\mathbf{n}^\top}{d}\right)\mathbf{x} \\
&\simeq \mathbf{K}\left(\mathbf{W}_{\text{rel}} - \frac{\mathbf{t}_{\text{rel}}\mathbf{n}^\top}{d}\right)\mathbf{K}^{-1}\widetilde{\mathbf{p}} \\
&\simeq \mathbf{H}\widetilde{\mathbf{p}}
\end{aligned} \tag{14}$$

Multi-view geometric consistency check. As mentioned in Sec. 4.2, we filter out inaccurate depth and normal through the multi-view geometric consistency check for the reference view. Specifically, for a pixel $\mathbf{p}$ in the reference view with estimated depth z, the depth z_g of the warped pixel $\mathbf{p}_g$ from reference view to a target view g can be induced by:

$$z_g\widetilde{\mathbf{p}}_g = \mathbf{K}(\mathbf{W}_{rel}\mathbf{K}^{-1}z\widetilde{\mathbf{p}} + \mathbf{t}_{rel}), \tag{15}$$

where $\widetilde{\mathbf{p}}$ and $\widetilde{\mathbf{p}}_g$ is the homogeneous coordinates of $\mathbf{p}$ and $\mathbf{p}_g$, $\mathbf{K}$ is the camera intrinsic, $[\mathbf{W}_{rel}, \mathbf{t}_{rel}]$ is the relative transformation from the reference view to the target view g. For each reference view, we choose four target views that are near the reference view and generate a geometric consistency mask $\mathbf{M}$ as:

$$\mathbf{M}(\mathbf{p}) = \begin{cases} 0, \text{ if } \left(\sum_{g=0}^{4}\beta\left(|z_g - z|/z\right)\right) < \tau \\ 1, \text{ otherwise.} \end{cases}, \beta(x) = \begin{cases} 0, \text{ if } x \geq \alpha \\ 1, \text{ otherwise.} \end{cases}, \tag{16}$$

where the pixel is retained if it appears in at least τ target views, and the absolute relative error between its warped depth and depth in the target view is within α.

Table 8: Ablation study on different propagation intervals.

Interval m	PSNR ↑	SSIM ↑	LPIPS ↓	Training time
10	**36.11**	**0.960**	**0.203**	78min
30	36.09	0.960	0.204	64min
50	36.08	0.960	0.204	56min
70	35.88	0.957	0.207	51min
90	35.86	0.957	0.208	**48min**

Table 9: Ablation study on different iterations of propagation for plane candidates.

Iteration u	PSNR ↑	SSIM ↑	LPIPS ↓	Training time
1	35.80	0.954	0.213	**53min**
2	36.01	0.958	0.209	55min
3	36.08	0.960	0.204	56min
5	36.10	**0.961**	0.204	59min
7	**36.11**	**0.961**	**0.203**	61min

B. More Ablation Studies

Number of iterations for propagation. As stated in implementation details of Section 5.1, we perform our proposed progressive propagation strategy every $m = 50$ training iterations where propagation is performed $u = 3$ times. The number of propagation iterations is determined by the parameters m and u, which we have experimentally evaluated and presented in Tables 8 and 9.

Table 8 illustrates that the quality of rendering improves as the interval between Gaussian propagation decreases, eventually converging when the interval reaches 50. Shorter intervals lead to more frequent propagation, allowing for more thorough optimization of the scene's geometry. However, an increase in the total number of propagation iterations also leads to higher time costs. Table 9 shows that an increase in propagation times u of plane candidates results in improved rendering quality, which stabilizes after reaching 3 iterations. Increasing the number of iterations allows candidate planes for a pixel to be propagated from more distant areas, enabling better error correction over larger areas. As the propagation of plane candidates is only one part of the Gaussian propagation module, the time cost increase associated with it is not as significant as the increase in Gaussian propagation times discussed in Table 8.

C. More Rendering Results

In this section, we present more results on Waymo (Fig. 8) and MipNeRF360 (Fig. 9) datasets. Compared to 3DGS, our method achieves more accurate rendering results and improved geometric structures, particularly the large-scale scenes on Waymo. Additionally, the results on MipNeRF360 reveal that 3DGS already optimizes accurate geometry, especially depth, for small-scale scenes with rich textures. Therefore, the improvements achieved by our method on the MipNeRF360 dataset are limited.

D. Results of Each Scene in Waymo and MipNeRF360

Table 10-15 presents the evaluation metrics for each scene in Waymo and MipNeRF360. For the MipNeRF360 dataset, we include the metrics from the original 3DGS paper (3DGS*), as well as the metrics obtained by rerunning 3DGS with new poses generated using COLMAP.

In the Waymo dataset, we achieve state-of-the-art (SOTA) rendering results for each scene, which significantly outperform the baseline 3DGS. In the MipNeRF360 dataset, we achieve improvements over 3DGS in indoor scenes(the first four rows), which contain many structured planes. In natural scenes with rich textures but lacking structured planes, our results are comparable to 3DGS. In these scenes, SfM techniques usually provide a high-quality point cloud for initialization and the simple clone and split densification strategies don't show a bottleneck. Besides, these scenes contain intricate fine structures that cover a few pixels, such as the grass and leaves, making it challenging to accurately estimate surface normals. Although ZipNeRF achieves the best rendering quality on the MipNeRF360 dataset, its rendering speed is significantly slower compared to ours (0.09 FPS for ZipNeRF compared to 108 FPS for our method).

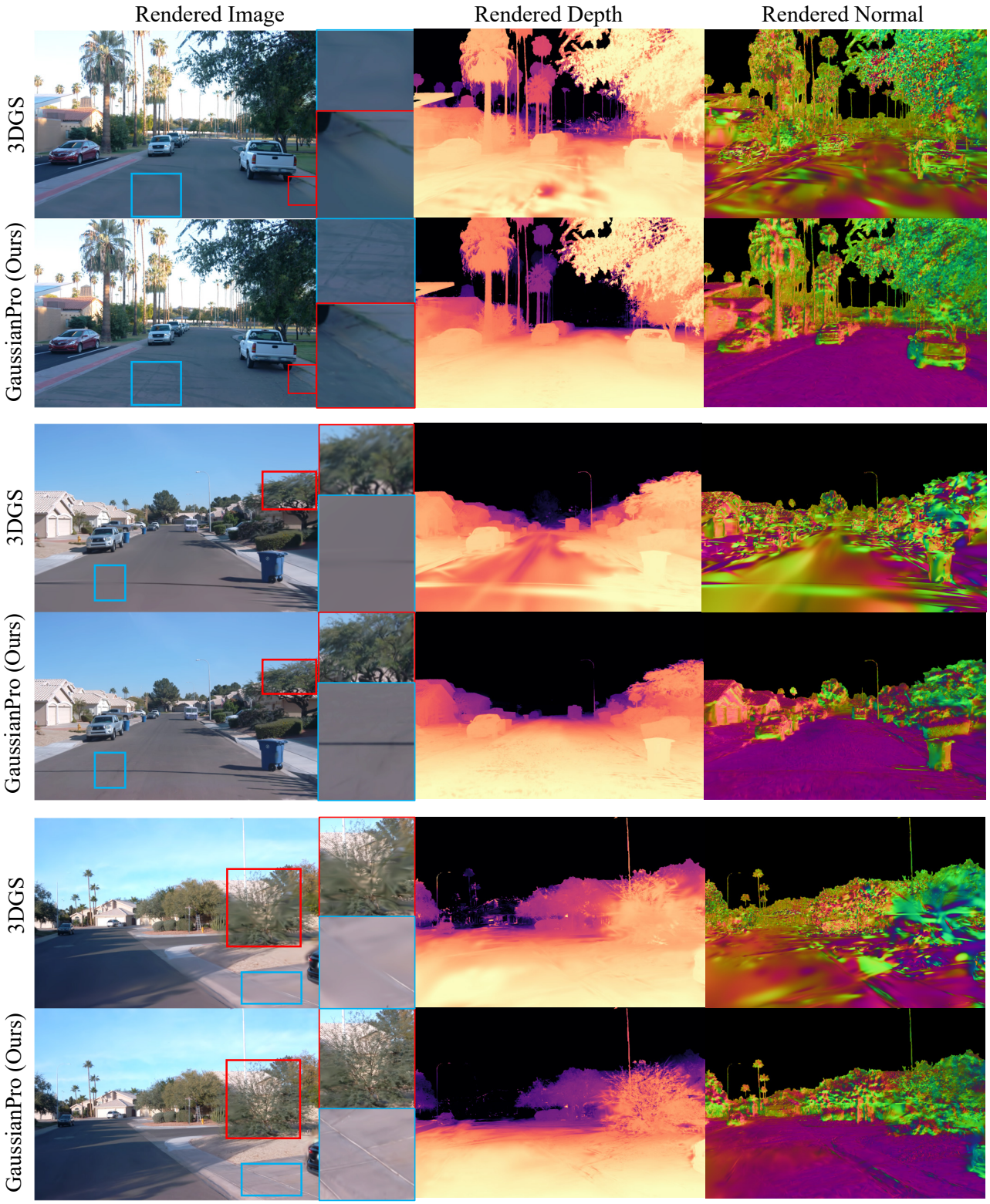


Figure 8: More rendering results on the Waymo dataset.

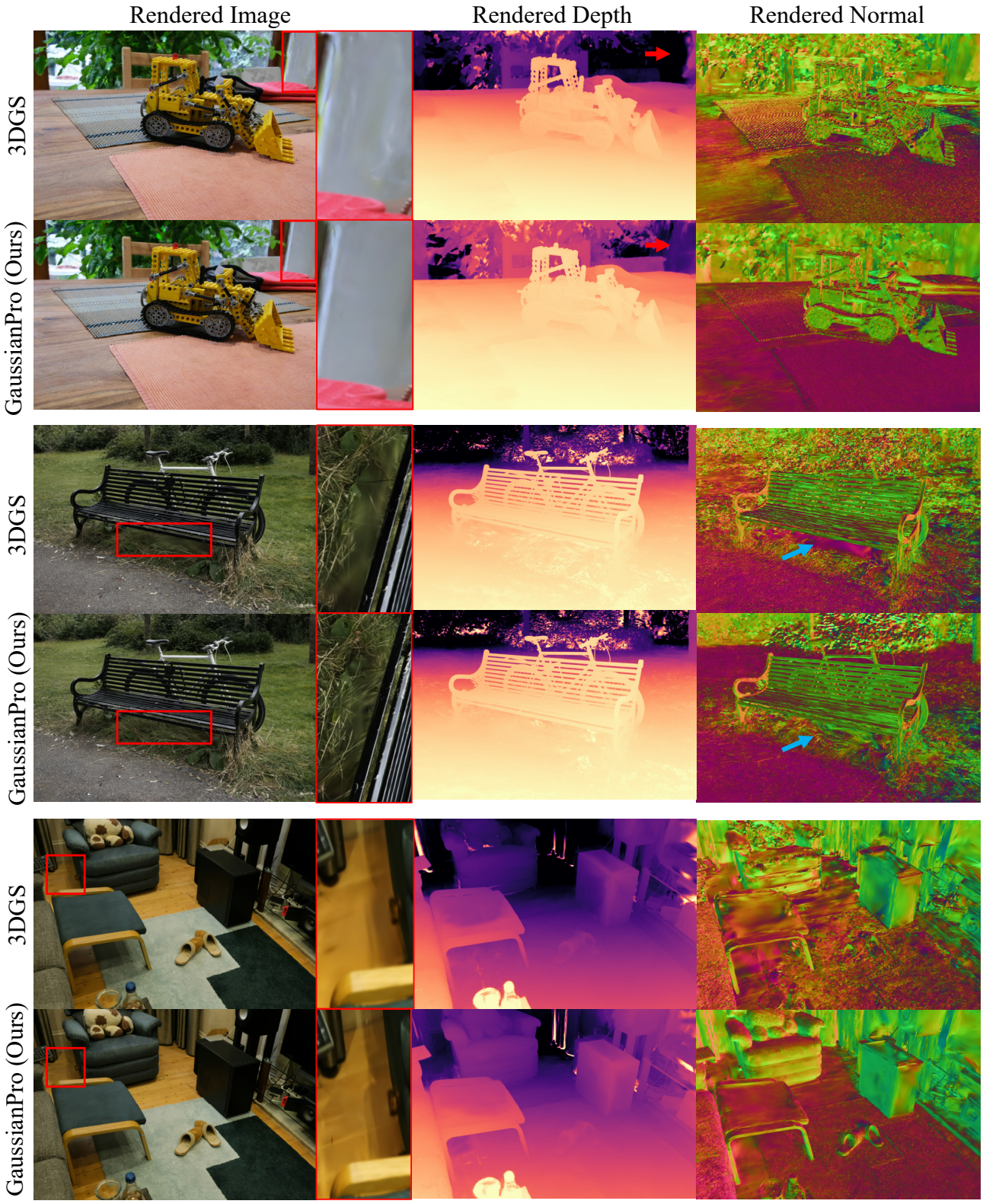


Figure 9: More rendering results on the MipNeRF360 dataset.

Table 10: SSIM results on the Waymo dataset.

Sequence	SSIM				
	NGP	Mip360	Zip	3DGS	Ours
Seg100613	0.823	0.927	0.954	0.947	**0.958**
Seg100275	0.917	0.952	0.957	0.950	**0.960**
Seg113792	0.905	0.934	0.947	0.933	**0.948**
Seg132384	0.935	0.948	**0.964**	0.957	0.963
Seg144248	0.852	0.762	0.896	0.938	**0.939**
Seg148697	0.856	0.885	0.924	0.913	**0.934**
Seg150623	0.928	0.937	0.963	0.960	**0.968**
Seg164701	0.855	0.909	0.923	0.917	**0.927**
Seg405841	0.899	0.924	0.927	0.931	**0.940**
Average	0.886	0.909	0.939	0.938	**0.949**

Table 11: PSNR results on the Waymo dataset.

Sequence	PNSR				
	NGP	Mip360	Zip	3DGS	Ours
Seg100613	33.67	31.07	36.55	35.71	**36.83**
Seg100275	32.38	34.27	**36.17**	35.05	36.08
Seg113792	32.63	32.67	**35.76**	33.26	34.87
Seg132384	33.03	32.90	**37.10**	35.52	36.54
Seg144248	28.92	21.17	32.12	34.54	**35.21**
Seg148697	27.95	27.20	30.88	29.75	**30.96**
Seg150623	33.19	31.16	37.59	36.94	**38.23**
Seg164701	27.35	29.40	29.90	29.38	**30.48**
Seg405841	29.70	30.93	31.93	31.63	**32.91**
Average	30.98	30.09	34.22	33.53	**34.68**

Table 12: LPIPS results on the Waymo dataset.

Sequence	LPIPS				
	NGP	Mip360	Zip	3DGS	Ours
Seg100613	0.237	0.201	0.178	0.209	**0.158**
Seg100275	0.262	0.257	**0.200**	0.233	0.204
Seg113792	0.295	0.289	**0.216**	0.239	0.233
Seg132384	0.234	0.200	**0.186**	0.218	0.190
Seg144248	0.364	0.396	0.303	0.239	**0.233**
Seg148697	0.323	0.230	0.185	0.224	**0.152**
Seg150623	0.246	0.214	0.191	0.214	**0.183**
Seg164701	0.292	0.275	0.194	0.212	**0.179**
Seg405841	0.277	0.292	0.195	0.218	**0.186**
Average	0.281	0.262	0.205	0.223	**0.191**

Table 13: SSIM results on the MipNeRF360 dataset.

Sequence	SSIM					
	NGP	Mip360	Zip	3DGS*	3DGS	Ours
Room	0.871	0.913	0.925	0.914	0.919	**0.927**
Counter	0.817	0.894	0.902	0.905	0.915	**0.916**
Kitchen	0.858	0.920	0.928	0.922	0.933	**0.935**
Bonsai	0.906	0.941	0.949	0.938	0.945	**0.952**
Bicycle	0.512	0.685	0.769	0.771	**0.810**	**0.810**
Flowers	0.486	0.583	**0.642**	0.605	0.603	0.598
Garden	0.701	0.813	0.860	0.868	**0.890**	**0.890**
Stump	0.594	0.744	**0.800**	0.775	0.769	0.763
Treehill	0.542	0.632	**0.681**	0.638	0.636	0.631
Average	0.699	0.792	**0.828**	0.815	0.824	0.825

Table 14: PSNR results on the MipNeRF360 dataset.

Sequence	PSNR					
	NGP	Mip360	Zip	3DGS*	3DGS	Ours
Room	29.69	31.63	**32.65**	30.63	31.71	31.98
Counter	26.69	**29.55**	29.38	28.70	29.06	29.17
Kitchen	29.48	32.23	**32.50**	30.32	31.57	31.60
Bonsai	30.69	33.46	**34.46**	31.98	32.69	33.05
Bicycle	22.17	24.37	25.80	25.25	**26.64**	26.60
Flowers	20.65	**21.73**	**22.40**	21.52	21.64	21.54
Garden	25.07	26.98	28.20	27.41	**28.77**	28.70
Stump	23.47	26.40	**27.55**	26.55	26.58	26.42
Treehill	22.37	22.87	**23.89**	22.49	22.30	22.21
Average	25.59	27.69	**28.54**	27.21	27.88	27.92

Table 15: LPIPS results on the MipNeRF360 dataset.

Sequence	LPIPS					
	NGP	Mip360	Zip	3DGS*	3DGS	Ours
Room	0.261	0.211	0.196	0.220	**0.192**	**0.192**
Counter	0.306	0.204	0.185	0.204	0.185	**0.175**
Kitchen	0.195	0.127	0.116	0.129	**0.113**	**0.113**
Bonsai	0.205	0.176	0.173	0.205	0.169	**0.163**
Bicycle	0.446	0.301	0.208	0.205	**0.194**	0.195
Flowers	0.441	0.344	**0.273**	0.336	0.346	0.342
Garden	0.257	0.170	0.118	**0.103**	0.107	0.107
Stump	0.421	0.261	**0.193**	0.210	0.236	0.244
Treehill	0.450	0.339	**0.242**	0.317	0.337	0.344
Average	0.331	0.237	**0.189**	0.214	0.209	0.208