GaussianPainter: Painting Point Cloud into 3D Gaussians with Normal Guidance

Jingqiu Zhou[2*], Lue Fan[2,3,4*], Xuesong Chen[2], Linjiang Huang[1†], Si Liu[1], Hongsheng Li[2,4]

[1]Beihang University
[2]Multimedia Laboratory, The Chinese University of Hong Kong
[3]Chinese Academy of Sciences
[4]Centre for Perceptual and Interactive Intelligence

Abstract

In this paper, we present GaussianPainter, the first method to paint a point cloud into 3D Gaussians given a reference image. GaussianPainter introduces an innovative feed-forward approach to overcome the limitations of time-consuming test-time optimization in 3D Gaussian splatting. Our method addresses a critical challenge in the field: the *non-uniqueness* problem inherent in the large parameter space of 3D Gaussian splatting. This space, encompassing rotation, anisotropic scales, and spherical harmonic coefficients, introduces the challenge of rendering similar images from substantially different Gaussian fields. As a result, feed-forward networks face instability when attempting to directly predict high-quality Gaussian fields, struggling to converge on consistent parameters for a given output. To address this issue, we propose to estimate a surface normal for each point to determine its Gaussian rotation. This strategy enables the network to effectively predict the remaining Gaussian parameters in the constrained space. We further enhance our approach with an appearance injection module, incorporating reference image appearance into Gaussian fields via a multiscale triplane representation. Our method successfully balances efficiency and fidelity in 3D Gaussian generation, achieving high-quality, diverse, and robust 3D content creation from point clouds in a single forward pass. A video is provided in our supplementary material for a more detailed explanation of our method.

1 Introduction

3D object generation has attracted increasing attention due to the potential to reduce human labor and professional software in the game, VR, and animation industries.

Over the past few years, implicit representations such as NeRF (Mildenhall et al. 2021; Xu et al. 2022b; Kondo et al. 2021) and Signed Distance Function(SDF) (Lionar et al. 2024; Park et al. 2019) have played major roles in this field. Recently, the emerging 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has become a hot trend because of its impressive rendering quality and efficiency. Different from implicit representations, 3D Gaussians can be viewed as a special type of point cloud with each point being decorated by rotation, scales, and colors. Such affinity between the Gaussians and point clouds raises a natural question: *Given reference information (e.g., reference images), can we transform a point cloud into a 3D Gaussian field in an efficient feed-forward manner?* Here the point clouds can be obtained from the existing 3D assets or point cloud generative models, such as Point-E (Nichol et al. 2022), PointFlow (Yang et al. 2019), and PointGrow (Sun et al. 2020). In this way, we try to take advantage of numerous existing 3D assets and the mature point cloud generators, and then efficiently *paint* point clouds into 3D Gaussians, as indicated by the paper title.

Compared with our method, existing methods (Chen, Wang, and Liu 2023; Tang et al. 2024; Haque et al. 2023) for 3D Gaussian generation are based on multi-step optimization or denoising, instead of a feed-forward paradigm. These methods are inefficient due to their multiple forward and backward iterations. Although the feed-forward Gaussian painting is more efficient, it is quite challenging for the following reason. 3D Gaussians are designed with a relatively large parameter space, unlike the previous point-based rendering methods (Lassner and Zollhofer 2021; Yifan et al. 2019) only utilizing spheres without rotations and anisotropic scales. Although such a large parameter space leads to the *non-uniqueness* of Gaussian fields. In other words, similar images can be rendered from totally different Gaussian fields, making it ambiguous for feed-forward models to predict Gaussian parameters in a single forward pass. This issue is also observed by recent work AGG (Xu et al. 2024), which turns to *isotropic* Gaussians (i.e., ignore rotations) to constrain the parameter space for more stable training and easier prediction. However, this solution weakens the capacity of 3D Gaussian and rendering quality.

In this paper, we propose GaussianPainter, which takes a point cloud and a reference image as input, generating 3D-Gaussian parameters for each point in a single forward pass. GaussianPainter contains two major components including (1) *Normal-guided Gaussian Painting* (Sec. 4.1) and (2) *Triplane-based Multiscale Appearance Injection* (Sec. 4.2). The first normal-guided Gaussian painting part addresses the issue of non-unique Gaussian fields by introducing *normal guidance*, where surface normals are treated as guidance to constrain the parameter space of Gaussians. More specifically, we first propose *Isotropic Normal Rendering* module to estimate a surface normal for each input point. The estimated normal is leveraged to define the rotations of the Gaussian centered at the point. After obtaining the rotations, the space of Gaussian parameters is compressed and constrained. It is much easier for the neural network to predict

*Equal contribution.
†Corresponding author.

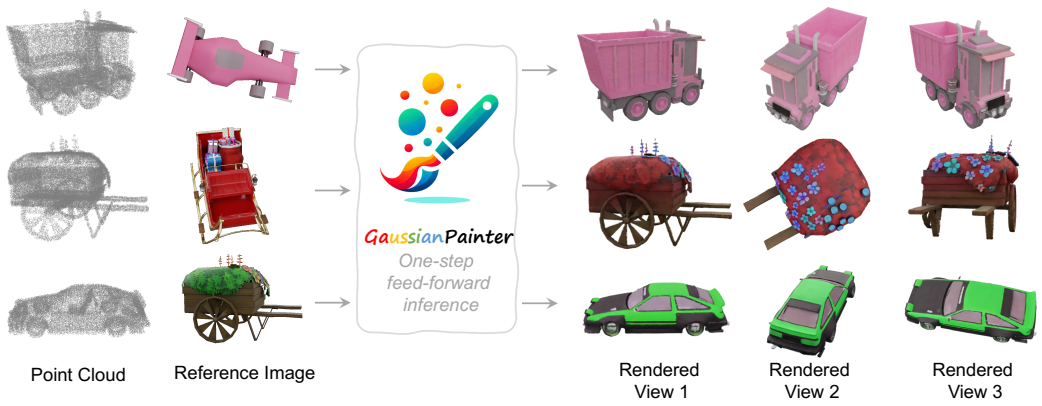

Figure 1: Given any reference image, the proposed GaussianPainter paints point clouds into 3D Gaussians in a feed-forward network.

the remaining parameters (i.e. scales, opacity, and colors). Moreover, the *Triplane-based Multiscale Texture Injection* part then injects the appearance from the reference image to Gaussians, for controllable Gaussian painting. In this module, the triplane representation narrows the modality gap between 2D reference images and 3D points, and the multi-scale feature enables high-fidelity appearance injection.

To summarize, our contributions are of three folds:

1. We treat the problem of 3D Gaussian generation as painting given point clouds into 3D Gaussians with a feed-forward neural network. This problem setup offers 3D object generation from a new perspective and leads to an efficient tool for creating diverse 3D content.
2. We analyze and effectively address the issue of non-unique Gaussian fields by introducing normal guidance on the training process, making the efficient feed-forward manner feasible.
3. Our method achieves state-of-the-art performance on the novel view synthesis task. In addition, we propose a novel cross-object appearance transfer task and our method demonstrates generalized transferring ability.

2 Related Work

2.1 3D Gaussian Splatting

3D Gaussians (Kerbl et al. 2023) is an explicit 3D representation that encodes rendering information in 3D anisotropic Gaussians. For each 3D Gaussian, it has the following properties: the mean $\mu \in \mathbb{R}^3$, the covariance $\Sigma \in \mathbb{R}^{3\times3}$, a scalar σ for opacity, and the view-dependent spherical harmonic coefficients $c \in \mathbb{R}^n$. The 3D Gaussian splitting has become a hot trend and is employed by many methods (Qian et al. 2023; Luiten et al. 2023; Yan et al. 2024; Guédon and Lepetit 2023; Gao et al. 2023; Zhou et al. 2023; Yugay et al. 2023) for various 3D tasks. Among them, a line of work (Chen, Wang, and Liu 2023; Tang et al. 2024; Li, Wang, and Tseng 2023; Liang et al. 2023; Liu et al. 2023b; Chung et al. 2023) focuses on the Gaussian generation from text instructions. Although these methods show promising results in generating objects, they usually take minutes to generate a single object due to the multi-step denoising process. In contrast, some methods (Zou et al. 2023; Xu et al. 2024) adopt feed-forward inference to create 3D objects from a single image, which is much faster than diffusion-based methods. However, the feed-forward manner faces a critical challenge of non-uniqueness when predicting 3D Gaussians, which will be presented in Sec. 3. In this paper, we are dedicated to boosting the performance of feed-forward methods by tackling this challenge.

2.2 3D Object Generation

The task of 3D object generation has always posed significant challenges. Earlier approaches (Li et al. 2021; Shih et al. 2020; Xu et al. 2022a) focused on limited view synthesis and achieved impressive 3D consistency. The recent advancements in 2D diffusion models lead to the development of a new diffusion-based paradigm (Poole et al. 2022; Wang et al. 2023, 2024) for 3D object generation. In this pipeline, the 2D diffusion model serves as a prior, and the 3D content is generated through optimization guided by this prior. This paradigm leads to many previous arts (Lin et al. 2023; Liu et al. 2023a, 2024; Melas-Kyriazi et al. 2023). Similar to the diffusion-based generation method in 3D Gaussian, these methods also face the inefficient multi-step denoising process, which again encourages us to explore the feed-forward methods.

3 Pilot Study: Understanding the Non-uniqueness of 3D Gaussians

As mentioned in Sec. 1, similar images can be rendered from totally different Gaussian fields, which makes it difficult for a generative model to predict high-quality 3D Gaussians. In this section, we conduct a pilot study to demonstrate this issue. Particularly, we first load a well-trained Gaussian field and fix the locations of each 3D Gaussian. We then randomly re-initialize and tune other Gaussian parameters, including rotation, scale, opacity, and spherical harmonics (SH) coefficients. This process is repeated multiple times, obtaining multiple Gaussian fields with the same locations but different other parameters. For each type of parameter, we define an Instability Score (IS) to measure its instability.

Assuming that we repeatedly train M Gaussian fields with N Gaussian locations in each field, the IS of a parameter

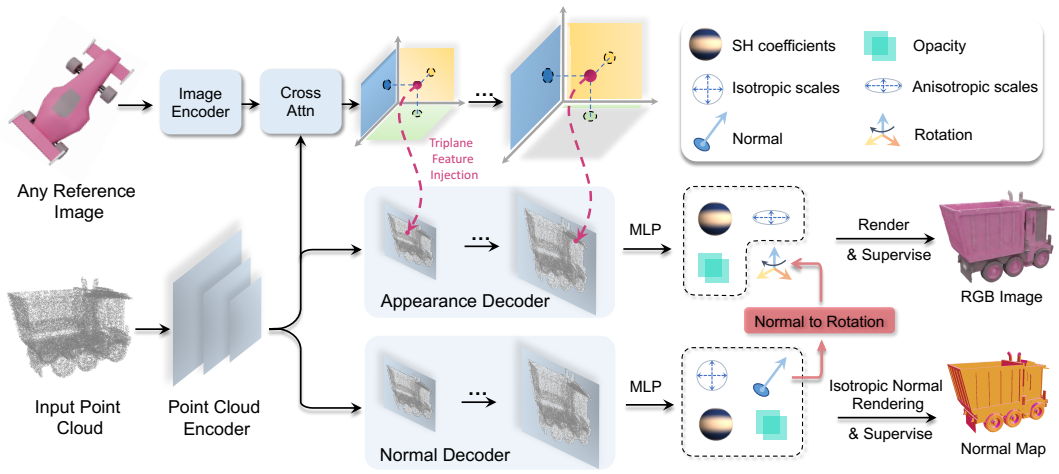

Figure 2: The overview of GaussianPainter. Its first major component is *Normal-guided Gaussian Painting*, consisting of Point Cloud Encoder, Appearance/Normal Decoder, and the following MLPs. This component directly operates on point clouds and predicts Gaussian parameters and normal for each point, presented in Sec. 4.1. The second major component is *Triplane-based Multiscale Appearance Injection*, which injects appearance information from reference image into the Appearance Decoder and guides Gaussian painting, presented in Sec. 4.2.

with C channels is calculated by following steps. (1) For each location and each channel, we calculate the standard variance of the M scalar values (one value per field). Thus, in total, we have $N \times C$ standard variances. We define their mean as *local instability score*. (2) Since different types of parameters have different scales, we calculate a global parameter scale for each type to normalize the instability score. For a certain parameter, its global parameter scale is defined as the standard variance of all the $M \times N \times C$ scalar values. (3) The final instability score is the ratio between the local instability and the global parameter scale. By definition, a large instability score indicates a large degree of non-uniqueness.

As shown by the statistics in Fig. 3, SH and opacity are relatively stable between different re-initializations. In contrast, scales and rotations are quite unstable, which are difficult to be predicted by generative models. Since scales are entangled with rotations, we consider the prediction of stable rotations as a key challenge to tackle, leading to our motivation to propose Normal Guidance to make the rotation prediction more stable and "unique".

4 Method

The proposed GaussianPainter takes a point cloud and a reference image as input, generating 3D-Gaussian parameters for each point in a single forward pass, including SH coefficients, opacity, scales, and rotations. The generated 3D Gaussians are expected to render high-quality images with a similar appearance provided by the reference image. GaussianPainter contains two major components: (1) *Normal-guided Gaussian Painting* (Sec. 4.1) and (2) *Triplane-based Multiscale Appearance Injection* (Sec. 4.2). The former introduces normal guidance to address the issue of non-unique Gaussians, and the latter injects appearance information from the reference image into 3D Gaussians. Fig. 2 illustrates the overall architecture of GaussianPainter.

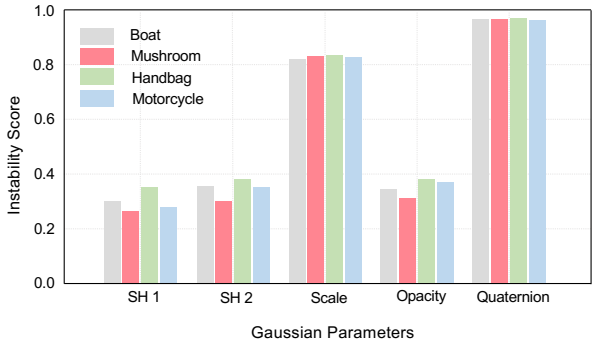

Figure 3: Demonstration of different Gaussian fields and Instability Score for different Gaussian parameters. SH 1 and SH 2 stand for the RGB component and the remaining component of harmonic coefficients, respectively.

4.1 Normal-guided Gaussian Painting

As discussed in the pilot study, the difficulty of Gaussian parameter prediction lies in the non-uniqueness and large parameter space of 3D Gaussians. In this section, we propose a normal guidance technique to constrain the parameter space for achieving more stable and better prediction.

Point Cloud Encoding. Given a point cloud (either from 3D scanning or generated by neural networks), we start by converting it to an occupancy grid with dimensions of $200 \times 200 \times 200$. We then adopt a 3D UNet for encoding the input occupancy. The 3D UNet is adapted from the conventional 2D UNet (Ronneberger, Fischer, and Brox 2015), where all 2D convolution layers are replaced with 3D sparse convolutions. In addition, all Batch Normalization layers are replaced with Layer Normalization to alleviate the unstable batch statistics caused by the sparsity of point clouds. The decoders of the 3D UNet upsamples the occupancy grid

back to its original resolution. Note that we have two UNet decoders sharing the same structure as shown by Fig. 2. The normal decoder predicts normals and other parameters within an isotropic Gaussian setting, while the appearance decoder predicts all parameters within an anisotropic setting for RGB rendering. We will present the details later.

Isotropic Normal Rendering. Based on the extracted occupancy features, we employ a multi-layer perceptron (MLP) to learn a directional vector for each occupied position. This vector is expected to point to the normal direction of the closest surface. In practice, some positions may deviate from the actual surface, due to the inherent noise in raw point clouds and the introduced extra occupied positions in the Point Cloud Encoding. As a result, defining precise normals for these off-surface positions is challenging. To address this issue, we refrain from directly supervising normal estimation in 3D space. Instead, we propose *Isotropic Normal Rendering* to render surface normals to a normal map and employ supervision on the 2D normal map. Specifically, an MLP takes the encoded occupancy feature as input and predicts Gaussian parameters for each occupied position, including a normal, isotropic scale, and opacity. Then we render the normals into a 2D normal map following

$$\boldsymbol{n} = \sum_{i=0}^{M} \boldsymbol{n}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{1}$$

Eq. (1) substitutes colors in the original 3DGS rendering with predicted normals. α is calculated from the predicted opacity and Gaussian density. Since the issue of non-uniqueness still exists during the normal prediction, the Gaussians in this module are defined as *isotropic*, to facilitate the prediction. Finally, the rendered normal map is supervised by L1 loss and SSIM loss. The properties of the proposed Isotropic Normal Rendering are worth further discussion:

- Leveraging normal rendering allows for the approximation of normals for all 3D positions, including those positions deviating from actual surfaces.
- Although these normals may not precisely match the true surface normals, they are sufficient to define rotations and constrain the space of feasible Gaussian parameters, fulfilling our requirements.
- The Gaussians here only serve the purpose of normal rendering. With the learned normals, a *new* set of Gaussians will be generated, which are anisotropic rather than isotropic, which will be discussed later.

Gaussian Prediction with Normal Guidance. Each occupied position is assigned a predicted normal $\boldsymbol{n}$, which is assumed to be a unit vector. We then demonstrate how to specify the Gaussian rotation $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ according to $\boldsymbol{n}$. Considering the three axes of the local coordinate system of a position, these axes are rotated from the canonical pose to pose $\boldsymbol{R}$, around a designated rotation axis $\boldsymbol{r}$. The rotation axis $\boldsymbol{r}$ is designated as

$$\boldsymbol{r} = \boldsymbol{n} \times \boldsymbol{z}, \tag{2}$$

where $\boldsymbol{z}$ is the unit vector along z-axis of the world coordinate (vertical direction). The rotation angle θ is the angle between $\boldsymbol{z}$ and $\boldsymbol{n}$. With rotation axis $\boldsymbol{r}$ and rotation angle θ, rotation $\boldsymbol{R}$ is calculated using Rodrigues' rotation formula, the details of which are omitted for brevity. Fig. 4 illustrates the rotation procedure.

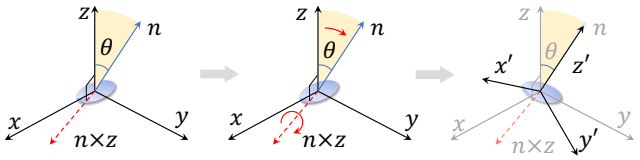

Figure 4: The illustration of converting the predicted normal to a rotation. The xyz-axes are rotated to $x'y'z'$-axes, which is considered as the rotation parameter of a Gaussian.

After obtaining the rotation $\boldsymbol{R}$, we leverage the appearance decoder and an MLP to predict the remaining Gaussian parameters based on the occupancy features. There are two notable differences from the Isotropic Normal Rendering module. 1) Here the MLP predicts a scale for each axis independently, so we have anisotropic Gaussians rather than isotropic Gaussians adopted in Normal Rendering. 2) The MLP predicts new opacity without reusing the opacity in the Normal Rendering module. With the newly predicted Gaussians, we render them into images and employ L1 loss and SSIM loss to optimize all trainable parameters in our model.

4.2 Triplane-based Multiscale Appearance Injection

To control the Gaussian painting, we propose Appearance Injection to inject the appearance information from reference images into the Gaussian painting process. This injection module features a multi-scale triplane design. The triplane representation bridges the dimensional gap between 3D Gaussians and 2D reference images, making the appearance injection aware of 3D structures. Moreover, the multi-scale design facilitates preserving the fidelity of appearance, which will be experimentally verified later.

Image Encoding with Multi-scale Triplane. Given a reference image, we first utilize DINOv2 (Oquab et al. 2023) to encode it into visual feature maps $\mathcal{F}$. Then we enhance $\mathcal{F}$ with 3D information by a simple cross attention with 3D occupancy features, formulated as

$$\tilde{\mathcal{F}} = \texttt{CrossAttn}(\mathcal{F}, [\mathcal{F}, \mathcal{O}], [\mathcal{F}, \mathcal{O}]), \tag{3}$$

where $\mathcal{O}$ is the flattened occupancy features from the last layer of the point cloud encoder and $[\ ]$ stands for concatenation in token dimension. In Eq. (3), $\mathcal{F}$ serves as attention queries and $[\mathcal{F}, \mathcal{O}]$ serves as keys and values. Here, we omit the notation for the flatten operation on $\mathcal{F}$ for clarity. We then denote the i-th plane of a triplane structure in the s-th scale as $\mathcal{T}_s^i$, where the superscript i stands for plane direction and $i \in \{\text{xy}, \text{yz}, \text{zx}\}$. $\mathcal{T}_s^i$ is constructed with the following rules

$$\mathcal{T}_s^i = \begin{cases} \texttt{UpLayer}_i(\tilde{\mathcal{F}}), & \text{if } s = 0 \\ \texttt{UpLayer}_i(\mathcal{T}_{s-1}^i), & \text{if } s > 0 \end{cases}, \tag{4}$$

where $\texttt{UpLayer}_i$ is an upsampling layer with stride 2 in 2D UNet (Rombach et al. 2022). The subscript i denotes the upsample layers for three directions, each with unique parameters.

Appearance Injection. With the multi-scale triplane features, we use each occupied 3D position to fetch the corresponding features in the three planes. We choose N_s layers in the appearance decoder of the UNet as *injected layers*, where N_s is the number of triplane scales. We denote an occupied position in the injected layer with scale s as $\boldsymbol{p}_s \in \mathbb{R}^3$. Its fetched feature from the triplane $\mathcal{T}_s$ can be denoted as

$$\boldsymbol{f}_s^{\boldsymbol{p}} = \texttt{CT}(\texttt{Ip}(\mathcal{T}_s^{\text{xy}}, \boldsymbol{p}_s^{\text{xy}}), \texttt{Ip}(\mathcal{T}_s^{\text{yz}}, \boldsymbol{p}_s^{\text{yz}}), \texttt{Ip}(\mathcal{T}_s^{\text{zx}}, \boldsymbol{p}_s^{\text{zx}})) \quad (5)$$

where CT and Ip stand for concatenation along channel dimension and bilinear interpolation, respectively. To inject the triplane feature into the occupancy grid, $\boldsymbol{f}_s^{\boldsymbol{p}}$ is projected by a linear layer to have the same channel dimension with the occupancy feature of $\boldsymbol{p}$. Then the projected $\boldsymbol{f}_s^{\boldsymbol{p}}$ is added to the occupancy feature. In this way, the multi-scale triplane features are injected into multiple layers of the point cloud decoder, guiding the rendering of Gaussians.

In conclusion, as shown in Fig. 2, the *Normal Decoder* and *Appearance Decoder* are pivotal to our method. The normal decoder predicts normals for occupied positions under an isotropic Gaussian setting. The Isotropic Normal Rendering module enables supervising normals on a 2D normal map. The appearance decoder integrates appearance information from multi-scale triplanes and normal guidance from the normal decoder, ultimately predicting the anisotropic Gaussians as the final output of GaussianPainter.

5 Experiments

5.1 Implementation Details

Point Cloud Encoder and Decoder. Based on the basic structure of 2D UNet (Rombach et al. 2022), the point cloud encoder progressively downsamples point clouds (i.e., occupancy grids after preprocessing). Each downsampling step halves the spatial size and doubles the channel size. In particular, we downsample the input occupancy grid by a factor of 16, while increasing the channel dimension from 32 to 512. Following the encoder, the normal decoder and appearance decoder gradually upsample the spatial size and decrease the channel dimension. Specifically, the two decoders recover the spatial size by $16\times$ and reduce channel dimension from 512 to 32.

Image Encoder and Triplanes. DINOv2 (Oquab et al. 2023) with ViT-based (Dosovitskiy et al. 2021) backbone is adopted as the image encoder. In our implementation, the DINOv2 encodes a 518×518 reference image into a 37×37 feature map. This feature map is further lifted into triplane features in four scales, including 37×37, 74×74, 144×144, and 288×288. Correspondingly, the channel dimensions of these triplanes are 512, 256,128, and 64.

Training Scheme. We take a single object as an example to demonstrate our training scheme. We first pre-render K views of this object. For each training iteration, we randomly sample a view as the reference image. Our model then predicts a Gaussian field and normals based on the reference image and point cloud. The predicted Gaussian field and normals are rendered into another random view for supervision. It is necessary to emphasize that solely using rendering loss without the need to train a Gaussian field for each object greatly simplifies the workflow. More details can be found in the supplementary materials.

5.2 Datasets, Evaluation, and Compared Methods

OmniObject3D (Wu et al. 2023) is a 3D dataset with over 6000 objects in 197 categories, which provides a sufficient number of blender-rendered views, surface normal maps, and point clouds (16384 points for each object). A subset of OmniObject3D which consists of 2437 objects across 73 categories is used as our total dataset. Within this subset, we sample a *validation split* for the evaluation of novel view synthesis, which contains two objects for each category.

Objaverse (Deitke et al. 2023) is a large-scale 3D dataset with more than 80k renderable 3D models for Blender. We select 10k high-quality assets from the LVIS split to construct our dataset. Unlike OmniObject3D, Objaverse does not provide point clouds, rendered images, or surface normals. To generate images with realistic light conditions, we render these objects with an HDRI environment image and cyclic engine in the Blender. Particularly, we render 100 posed images for each object, and the corresponding surface normal maps are saved accordingly. A surface sampling strategy (Haggerty 2019) is adopted to extract point clouds. Because objects in Objaverse are far more complicated than OmniObject3D, we sample 32768 points for each object in our dataset.

Evaluation. We conduct both quantitative and qualitative evaluations. For better understanding, we assume each object has K ground-truth views. For the evaluation of the i-th view, we randomly choose another view from the remaining $K-1$ views as the reference to paint the point cloud into Gaussians. Then the generated Gaussians are rendered into the i-the view to evaluate the quality of the i-th rendered view. The process above is repeated K times with a different reference view and rendering view each time, which avoids the bias caused by a fixed choice of reference view. The overall evaluation result of each object is the average of results in K views. For qualitative evaluation of an object, we can specify any image as the reference, which is not limited to a certain rendered view of the object.

Compared Methods. We compare our method with the two baseline methods AGG (Xu et al. 2024) and TriplaneGaussian (Zou et al. 2023). AGG is a two-stage method that first generates a reasonable point cloud according to the input image and then predicts Gaussian parameters based on the generated point cloud. Given that AGG is not open-sourced, for a fair comparison, we substitute its generated point clouds with ground-truth point clouds and re-implement their pipeline, achieving similar results to their official results. The most distinctive difference between our GaussianPainter and AGG lies in that AGG simply predicts isotropic Gaussians, overlooking the challenge of non-uniqueness presented in Sec. 3.

TriplaneGaussian is another related work to generate 3D Gaussians in a feed-forward manner. The authors provide an official inference script and pretrained weights for the Objaverse dataset. For a fair comparison, we also substitute its generated point clouds with ground-truth point clouds during the inference. It is worth emphasizing that TriplaneGaussian also overlooks the challenge of non-uniqueness in Sec. 3. In the following sections, we will show that handling this challenge with normal guidance significantly boosts the results.

5.3 Main Results

Novel View Synthesis. In this subsection, we conduct novel view synthesis to quantitatively evaluate GaussianPainter, following the evaluation protocol in Sec. 5.2. We compare our GaussianPainter with AGG (Xu et al. 2024) and TriplaneGaussian (Zou et al. 2023) in terms of PSNR, SSIM, and LPIPS. In our experiments, we center-cropped the ground-truth image and the rendered image to a size of 600×600. In this way, we significantly reduced the information-less background region so that the evaluation metric could better reflect the rendering quality. The results listed in Table 1 are obtained with the OmniObject3D valid split in Sec. 5.2. Our method achieves superior performance.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
TriplaneGaussian (Zou et al. 2023)	25.8	0.881	0.191
AGG (Xu et al. 2024)	28.4	0.914	0.177
GaussianPainter (ours)	30.9	0.945	0.134

Table 1: Comparison with previous methods on the OmniObject3D for Novel View Synthesis.

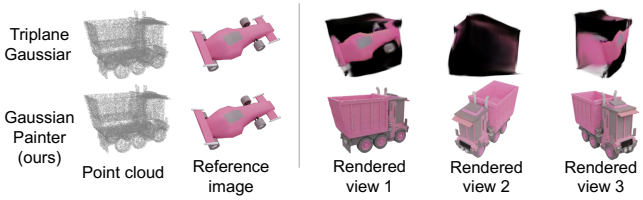

Figure 5: Qualitatively comparison for the task of cross-object appearance transfer on Objaverse. Our method demonstrates the superiority in transferring reasonable appearance to another object.

Cross-object Appearance Transfer. Beyond synthesizing novel views, GaussianPainter shows exciting generalization in transferring the appearance between two quite different objects. Note that cross-object samples have never been seen by GaussianPainter during training. Unlike the novel view synthesis, cross-object transfer requires understanding the semantics of the object parts, in addition to simply learning the mapping from the 2D reference image to the 3D space. As Fig. 5 shows, in the cross-object setting, TriplaneGaussian predicts a large portion of black 3D Gaussians. We can also find the shape of the formula racing car persists in the rendered truck (first row, rendered view 1). This phenomenon reveals that TriplaneGaussian may only learn correspondences instead of the correct semantics, resulting in simply "mapping" 2D appearance into 3D Gaussians. More results are shown in the supplementary materials.

5.4 Ablation Study and Analysis

Effectiveness of Normal Guidance To validate the effectiveness of Normal Guidance, we delete the whole normal prediction branch and add rotation predictions in another branch, which we name as rotation prediction model. However, training the rotation prediction model from scratch is quite unstable and the model is hard to be well-converged. Thus, we utilize the pretrained weights of our normal-guided model as initialization and fine-tune the rotation prediction model. Table 2 demonstrates the results, which leads to the following findings.

Prediction Target	PSNR ↑	SSIM ↑	LPIPS↓
Rot. (from scratch)	25.8	0.899	0.211
Rot. (fine-tuned)†	29.5	0.924	0.156
w/o. Rot. (isotropic scale)	28.4	0.914	0.177
Normal-guided Rot. (ours)	30.9	0.945	0.134

Table 2: Comparison between normal guidance and other rotation prediction strategies. Rot. stands for rotation. †: this model is fine-tuned to predict rotation from a model trained with normal guidance.

Design Choice	PSNR ↑	SSIM ↑	LPIPS ↓	#Param.
Single-scale triplane	29.2	0.926	0.155	511 M
Multi-scale triplane	30.6	0.930	0.142	549 M
Attention-only†	27.4	0.905	0.198	458 M
Full model	30.9	0.945	0.134	591 M

Table 3: Effectiveness of multi-scale triplane and cross-modal attention (Eq. (3)). The single-scale and multi-scale experiments do not adopt cross-modal attention. The full model integrates multi-scale triplane and cross-modal attention. †: The attention-only setting does not obtain information from triplanes. It basically follows Eq. (3) to interact with images but uses flattened 3D occupancy features as the attention queries.

- The proposed normal-guided model achieves the best performance, demonstrating the effectiveness of normal guidance.
- The rotation prediction model trained from scratch has the worst results, which again verifies the key challenge caused by unstable rotations presented in Sec. 3.
- Normal-guided supervision enables the model to predict stable rotations even without explicitly converting normals to rotations, as demonstrated by the fine-tuned model.
- Predicting isotropic Gaussians without rotation leads to stable training but mediocre performance, which is the solution adopted by AGG (Xu et al. 2024).

We also provide a qualitative comparison between these settings using the objects from OmniObject3D, shown in Fig. 6.

The Effectiveness of Multiscale Triplane and Cross-modal Attention. The multiscale characteristic and cross-model attention (Eq. (3)) are two crucial techniques for building effective triplane representation. To support our claim, we develop three model variants and compare their performance with the Novel View Synthesis on the OmniObject3D dataset. The results are listed in Table 3.

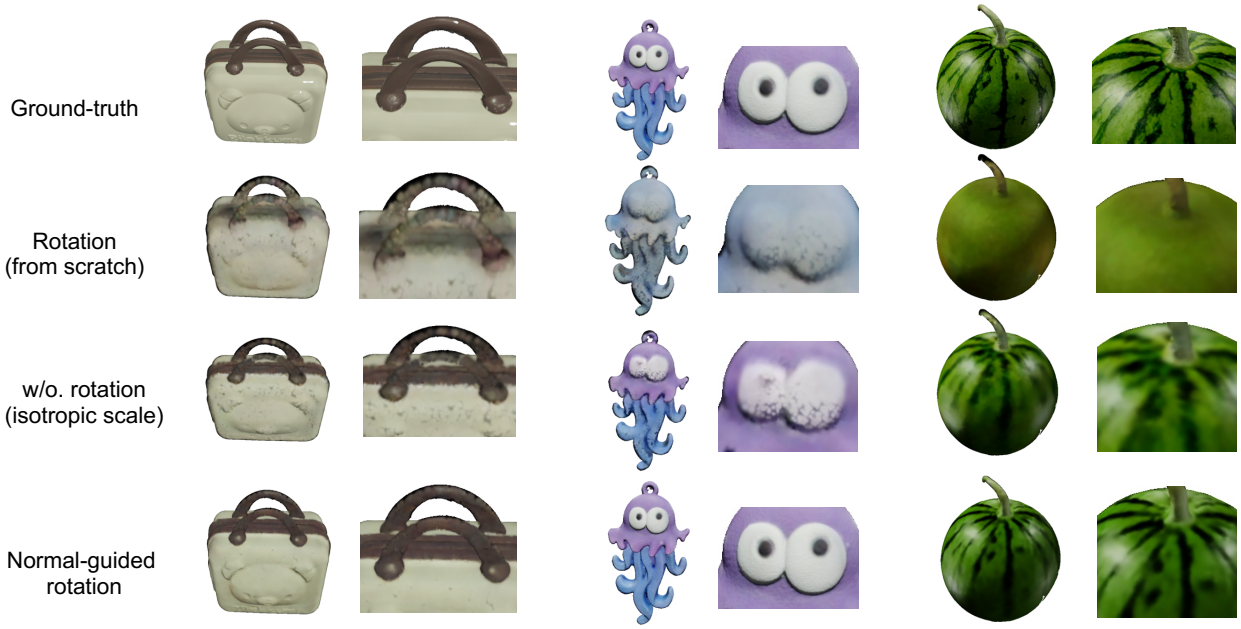

Figure 6: Qualitative comparison between different rotation prediction strategies, using the OmniObject3D validation split constructed in Sec. 5.2. To maintain a consistent appearance among different settings for easier comparison, we let the reference view and rendered view be the same. In this setting, the model could produce very precise appearance offered by the reference view (i.e., the first row in this figure).

Noisy Points	0%	10%	30%	50%	70%	90%
PSNR	27.5	27.1	26.9	25.9	24.3	23.7

Table 4: Impact of noisy point cloud.

Performence on generated point clouds and noisy point clouds. We further conduct experiments to verify how well our method adapts to generate point clouds. We utilize a point cloud generator (Zou et al. 2023) to generate point clouds, which are then painted into renderable Gaussians with our GaussianPainter . As shown in Fig. 7 (2nd column), our model can directly generalize to generated point clouds. After finetuning our model on the point cloud generator, the results are further improved. More rigorously, we analyze the robustness of our method against noisy point clouds. These point clouds are generated by perturbing a portion of the GT point clouds with Gaussian noise. Table 4 shows that our model effectively handles highly noisy input point clouds.

6 Conclusion and Future Work

In this study, we present GaussianPainter, an innovative method that effectively transforms point clouds into high-quality Gaussian splatting fields in a feed-forward manner, using any reference image with appearance information. Our approach tackles the non-uniqueness challenge in fitting an anisotropic Gaussian splatting field by introducing normal guidance to stabilize the rotation parameter, significantly reducing the fitting difficulty. Additionally, we introduce a multi-scale triplane representation to better preserve the appearance fidelity of the reference image. GaussianPainter shows the capacity of high-quality, diverse, and robust 3D content creation from point clouds in a single pass.

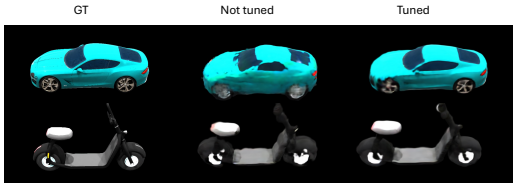

Figure 7: The results of GaussianPainter on generated point cloud for novel view synthesis. The reference image is sampled from a different view.

However, GaussianPainter has not been applied to large-scale scene-level point clouds. Our future research endeavors involve scaling the size of our training dataset, scene-level painting, and better controllability by text instructions and intricate appearance.

Acknowledgements

The project is funded by National Key R&D Program of China Project 2022ZD0161100, by Shanghai Artificial Intelligence Laboratory (Grant No. 2022ZD0160104), by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Commission (ITC)'s InnoHK, by General Research Fund of Hong Kong RGC Project 14204021. of CPII under the InnoHK.

References

Chen, Z.; Wang, F.; and Liu, H. 2023. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*.

Chung, J.; Lee, S.; Nam, H.; Lee, J.; and Lee, K. M. 2023. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*.

Deitke, M.; Schwenk, D.; Salvador, J.; Weihs, L.; Michel, O.; VanderBilt, E.; Schmidt, L.; Ehsani, K.; Kembhavi, A.; and Farhadi, A. 2023. Objaverse: A universe of annotated 3d objects. In *CVPR*, 13142–13153.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Gao, J.; Gu, C.; Lin, Y.; Zhu, H.; Cao, X.; Zhang, L.; and Yao, Y. 2023. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*.

Guédon, A.; and Lepetit, V. 2023. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*.

Haggerty, D. 2019. trimesh. https://trimesh.org/. Accessed: 2019-12-08.

Haque, A.; Tancik, M.; Efros, A. A.; Holynski, A.; and Kanazawa, A. 2023. Instruct-nerf2nerf: Editing 3d scenes with instructions. *ICCV*.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).

Kondo, N.; Ikeda, Y.; Tagliasacchi, A.; Matsuo, Y.; Ochiai, Y.; and Gu, S. S. 2021. Vaxnerf: Revisiting the classic for voxel-accelerated neural radiance field. *arXiv preprint arXiv:2111.13112*.

Lassner, C.; and Zollhofer, M. 2021. Pulsar: Efficient sphere-based neural rendering. In *CVPR*, 1440–1449.

Li, J.; Feng, Z.; She, Q.; Ding, H.; Wang, C.; and Lee, G. H. 2021. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *ICCV*, 12578–12588.

Li, X.; Wang, H.; and Tseng, K.-K. 2023. Gaussiandiffusion: 3d gaussian splatting for denoising diffusion probabilistic models with structured noise. *arXiv preprint arXiv:2311.11221*.

Liang, Y.; Yang, X.; Lin, J.; Li, H.; Xu, X.; and Chen, Y. 2023. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *arXiv preprint arXiv:2311.11284*.

Lin, C.-H.; Gao, J.; Tang, L.; Takikawa, T.; Zeng, X.; Huang, X.; Kreis, K.; Fidler, S.; Liu, M.-Y.; and Lin, T.-Y. 2023. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 300–309.

Lionar, S.; Xu, X.; Lin, M.; and Lee, G. H. 2024. Nu-mcc: Multiview compressive coding with neighborhood decoder and repulsive udf. *NeurIPS*, 36.

Liu, M.; Xu, C.; Jin, H.; Chen, L.; Varma T, M.; Xu, Z.; and Su, H. 2024. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *NeurIPS*, 36.

Liu, R.; Wu, R.; Hoorick, B. V.; Tokmakov, P.; Zakharov, S.; and Vondrick, C. 2023a. Zero-1-to-3: Zero-shot One Image to 3D Object. In *ICCV*.

Liu, X.; Zhan, X.; Tang, J.; Shan, Y.; Zeng, G.; Lin, D.; Liu, X.; and Liu, Z. 2023b. Humangaussian: Text-driven 3d human generation with gaussian splatting. *arXiv preprint arXiv:2311.17061*.

Luiten, J.; Kopanas, G.; Leibe, B.; and Ramanan, D. 2023. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*.

Melas-Kyriazi, L.; Laina, I.; Rupprecht, C.; and Vedaldi, A. 2023. Realfusion: 360deg reconstruction of any object from a single image. In *CVPR*, 8446–8455.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Nichol, A.; Jun, H.; Dhariwal, P.; Mishkin, P.; and Chen, M. 2022. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 165–174.

Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Qian, Z.; Wang, S.; Mihajlovic, M.; Geiger, A.; and Tang, S. 2023. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. *arXiv preprint arXiv:2312.09228*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR*, 10684–10695.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 234–241. Springer.

Shih, M.-L.; Su, S.-Y.; Kopf, J.; and Huang, J.-B. 2020. 3d photography using context-aware layered depth inpainting. In *CVPR*, 8028–8038.

Sun, Y.; Wang, Y.; Liu, Z.; Siegel, J.; and Sarma, S. 2020. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *WACV*, 61–70.

Tang, J.; Ren, J.; Zhou, H.; Liu, Z.; and Zeng, G. 2024. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *ICLR*.

Wang, H.; Du, X.; Li, J.; Yeh, R. A.; and Shakhnarovich, G. 2023. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 12619–12629.

Wang, Z.; Lu, C.; Wang, Y.; Bao, F.; Li, C.; Su, H.; and Zhu, J. 2024. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *NeurIPS*, 36.

Wu, T.; Zhang, J.; Fu, X.; Wang, Y.; Ren, J.; Pan, L.; Wu, W.; Yang, L.; Wang, J.; Qian, C.; et al. 2023. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *CVPR*, 803–814.

Xu, D.; Jiang, Y.; Wang, P.; Fan, Z.; Shi, H.; and Wang, Z. 2022a. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *ECCV*, 736–753. Springer.

Xu, D.; Yuan, Y.; Mardani, M.; Liu, S.; Song, J.; Wang, Z.; and Vahdat, A. 2024. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint arXiv:2401.04099*.

Xu, Q.; Xu, Z.; Philip, J.; Bi, S.; Shu, Z.; Sunkavalli, K.; and Neumann, U. 2022b. Point-nerf: Point-based neural radiance fields. In *CVPR*, 5438–5448.

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

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 4541–4550.

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

Yugay, V.; Li, Y.; Gevers, T.; and Oswald, M. R. 2023. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*.

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

Zou, Z.-X.; Yu, Z.; Guo, Y.-C.; Li, Y.; Liang, D.; Cao, Y.-P.; and Zhang, S.-H. 2023. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147*.