TUM Data Innovation Lab

Munich Data Science Institute (MDSI)

Technical University of Munich

&

TUM Computer Vision Group

&

Oxford Visual Geometry Group

Final Report for the Project:

Object-Centric 3D Reconstruction and Decomposition

Authors	Wenbo Ji, Michael Neumayr, Nina Kirakosyan, Filip Skubacz
Mentors	Dr. Yan Xia, Dr. Chuanxia Zheng
TUM Mentor	Dr. Yan Xia
Project lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

July 2025

Abstract

This report explores advanced techniques for 3D object reconstruction and generation using 3D Gaussian Splatting (3DGS) representations and feed-forward models, with applications in augmented reality, virtual reality, game development, robotics, and vision-based navigation. We focus on three interconnected topics in object-centric 3D reconstruction and decomposition.
First, we address the reconstruction of occluded objects in 3D. Using 3DGS, we infer the complete forms of partially visible objects from single images while addressing the challenges of occlusion and geometric coherence. Second, we investigate efficient 3D object generation by reimplementing and modifying recent generative adversarial network (GAN) architectures. These architectures incorporate geometric priors or hierarchical regularization to align Gaussian representations with accurate object geometry. Finally, we investigate scene decomposition from sparse views. A multi-view transformer and a 2D U-Net predict depth maps and Gaussian parameters, enabling efficient scene reconstruction and detailed object segmentation based on semantic properties.
This work demonstrates the versatility and potential of 3DGS and feed-forward models in 3D computer vision, with possible applications ranging from robotics to entertainment and virtual design.

Acknowledgments

First and foremost, we would like to express our heartfelt gratitude to our mentors, Dr. Yan Xia and Dr. Chuanxia Zheng. Their generous support, insightful guidance, and constructive feedback were instrumental throughout the project. We greatly appreciate their dedication and the fascinating, challenging research topics they proposed. Special thanks go to our project supervisor, Dr. Ricardo Acevedo Cabra, for his valuable support and project coordination, which made our work possible. Additionally, we thank Dr. Yan Xia and the system administrators of the TUM Computer Vision Group for providing the computing resources that were crucial to the successful completion of our project. We would also like to express our deepest gratitude to Prof. Dr. Massimo Fornasier and everyone else involved in making this project possible. Special thanks go to the legal teams, advisers, technical and ethical evaluation experts, secretaries, financial experts, professors, institute directors, lab managers, mentors, and human resources personnel. Their dedication and hard work behind the scenes were crucial to the success of our project.

Contents

1 Introduction

3D object reconstruction, generation, and decomposition have broad applications in embodied artificial intelligence, in which agents learn through interaction and exploration to solve challenging tasks in their environments. The study of embodied agents both challenges us to build intelligent systems and helps us understand how intelligence emerges through interaction with an environment. In this project, we explore several directions in the field, including occluded-object reconstruction, 3D object generation, and scene decomposition. These are interconnected and complementary research areas. Reconstructing occluded objects is an important step toward understanding and enhancing constrained environments. Fast feed-forward frameworks for 3D object generation could support both virtual-environment creation and data generation for related 3D tasks, including occluded-object reconstruction. Moreover, a better understanding of the environment through scene decomposition enables downstream applications such as navigation in unknown environments and manipulation of surrounding objects. We hope that our research will contribute to efficient, high-quality solutions for embodied tasks.

2 Background

A neural radiance field (NeRF) [27] samples points along each ray and queries a multilayer perceptron (MLP) to obtain the corresponding colors and opacities of a 3D scene; this process can be viewed as backward mapping (ray tracing). In contrast, 3D Gaussian Splatting (3DGS) [22] projects all 3D Gaussians into image space and rasterizes them in parallel, which can be viewed as forward mapping (splatting and rasterization). A scene is typically represented by millions of 3D Gaussians. Each Gaussian is parameterized by its mean (the 3D position), an anisotropic covariance, opacity α, and spherical harmonics (SH) coefficients.

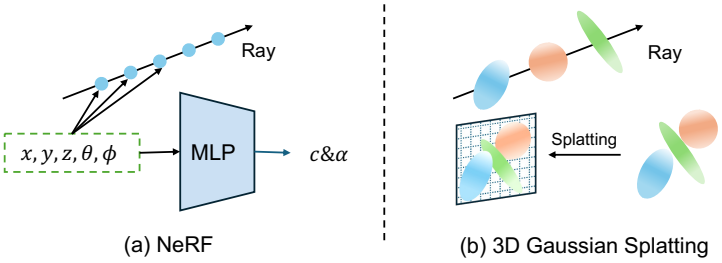


Figure 1: Comparison of NeRF and 3DGS.

In the original setup, a reconstruction objective over 2D views optimizes Gaussians initialized from a structure-from-motion (SfM) point cloud. Their initial scales are derived from the mean distance to the three nearest neighbors. An efficient differentiable tile rasterizer facilitates optimization from a set of images. In addition, an adaptive density-control module prunes, clones, and splits Gaussians, progressively increasing their number to capture finer scene details.

3 Reconstructing Occluded Objects in 3D

Chapter author: Filip Skubacz

Humans can understand their physical environment beyond what is directly visible; by leveraging prior knowledge, they can infer the 3D structure of a scene containing partially visible objects. In contrast, state-of-the-art machine learning models excel at superficial scene interpretation but struggle with implicit details that require a deeper understanding [1]. Our objective is to train a computer to recognize partially visible objects from a single image and reconstruct their complete three-dimensional forms. Such a model could subsequently be applied to downstream vision tasks and, by extension, to robotics, where it could significantly improve planning and navigation [48, 26]. We aim to build a simple, fast feed-forward model.

This task is challenging because it combines two difficult problems. First, by definition, the image does not directly provide the missing information, resulting in ambiguity. Consequently, the model has to take an educated guess to infer the missing details. This raises important questions about inference under insufficient information and about generalization.

First, we summarize the relevant concepts and review related research. Next, we introduce the methodology and experiments, followed by a brief discussion.

3.1 Background

In this section, we provide background for our method and review related work.
3DGS, introduced in Section 2, is an explicit 3D volume-rendering representation known for its speed. Traditionally, a Gaussian representation is reconstructed from multi-view inputs to model a 3D scene. Occlusions can be problematic in this setting, and methods such as [24] seek to address them.

Sparse/Single-View 3D Reconstruction Various 3D representations are utilized in computer vision, each suited to different use cases. For single-view 3D reconstruction, most approaches learn to predict Gaussian parameters or a NeRF by rendering from a new camera position and comparing the result with ground truth, a process known as novel view synthesis (NVS). Recent influential work in this domain includes Zero-1-to-3 [25], the Large Reconstruction Model (LRM) [16] and extensions such as Real-3D [18], Stable Video 3D (SV3D) [41], and the simple, fast Splatter Image method [38].
Sparse-view and single-view reconstruction methods must inherently hallucinate plausible details in regions where data is missing, such as the rear of an object. In this regard, reconstruction under occlusion is conceptually similar: a model capable of handling one type of missing information could potentially learn to manage the other. However, in the case of occlusions, the input data is even sparser, and uncertainty is greater. We use this as motivation for the *naïve* approach discussed later in Section 3.3.

Existing Occlusion Reconstruction Models Works such as [30] and [26] focus on so-called *amodal completion*. Amodal perception is the task of understanding more than what is fully visible in an image, such as inferring the hidden parts of partially occluded objects, a capability that humans possess [1]. The aforementioned models build towards this goal by fully completing an object in 2D, not just its shape. [30] employs an off-the-shelf diffusion model conditioned on partial objects, whereas [26] divides the task into two stages: a diffusion model conditioned on partial objects maps the input to a latent-space representation, which a separately trained decoder translates into image space. Both approaches rely on computationally expensive diffusion-based inference rather than feed-forward prediction [30, 26]. Moreover, the objects are completed in 2D, while 3D reconstruction is treated as a downstream task. Our experiments attempt to bypass this intermediate step and develop a straightforward model capable of understanding complete 3D objects.

3.2 Method

In the following section, we introduce various independent modifications and additions aimed at altering the model to deal with occlusions. We build on the Splatter Image framework, a simple U-Net architecture that predicts a Gaussian for every pixel of the input image [38]. For brevity, we denote the original Splatter Image model as SP_0.
We hypothesize that occlusions might result in (a) localized issues in the occluded regions of the object or (b) degradation of the entire reconstruction. To address (a), we propose symptom-focused strategies that guide the learning process in a manner we believe to be beneficial. In contrast, fundamental changes to the architecture or learning process can address both (a) and (b) simultaneously. Theoretically, these approaches could be applied to other single-view reconstruction methods.

Modifications All experiments use an extension of *multi-category ShapeNet* [6] with 64×64 renderings; the low resolution reduces training costs while remaining faithful to the problem statement. Most objects lack rich textures. Combined with the low resolution, this could hinder generalization to in-the-wild images.
Because our task differs from the original task, we modify the dataset. We randomly obstruct dataset objects with other dataset objects using randomized transformations, keeping occlusion within specified minimum and maximum bounds, as illustrated in Figure 2. The ground truth consists of the fully visible object x, the occluding object y, and their composition z, in which the object to be reconstructed may be only partially visible. Each is a red–green–blue–alpha (RGBA) image normalized to $[0, 1]$. The composition z includes the occlusion mask in its alpha channel; for real-world images, this mask would first need to be predicted.
The following loss function combines an $\mathcal{L}_2$ reconstruction term, Learned Perceptual Image Patch Similarity (LPIPS), and a generative adversarial network (GAN) objective to train both the generator G, which predicts the Gaussians, and the discriminator D:

$$\overbrace{\lambda_1 \mathbb{E}\left[w_{x,y}||x - G(z)||_2\right] + \lambda_2 \mathbb{E}\left[\mathrm{LPIPS}(x, G(z))\right]}^{\text{Reconstruction loss}} + \underbrace{\mathbb{E}\left[\log D(x)\right] + \mathbb{E}\left[\log(1 - D(G(z)))\right]}_{\text{GAN loss}},$$

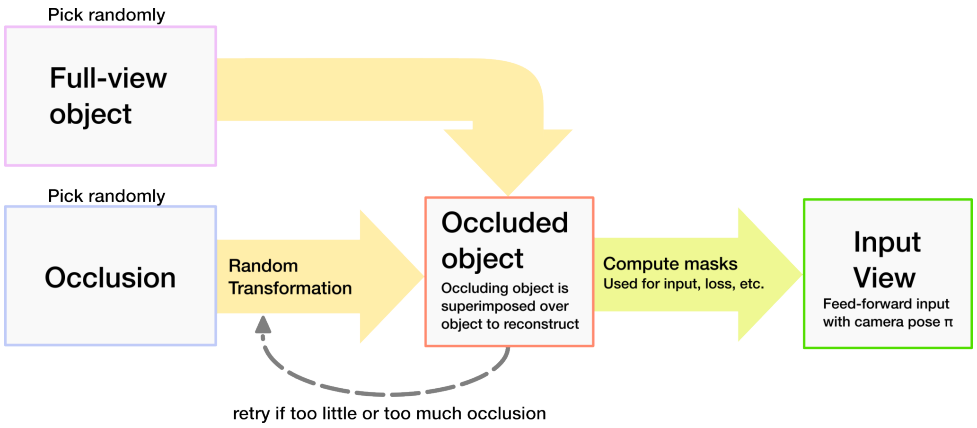


Figure 2: Dataset generation for the input view. The ground truth used in the loss computation remains unchanged.

(1)

where x is a fully visible object, z is an occluded object, and $w_{x,y}$ is a loss-weight vector that depends on x and y. The discriminator maximizes Equation 1 with $\lambda_1 = \lambda_2 = 0$ and learns to distinguish ground-truth RGB images from reconstructed ones. An additional, equally weighted discriminator evaluates the shape, represented by the mask. This concept is adapted from [13]. SP_0 was trained solely with the reconstruction loss.
Optionally, the different regions can be weighted differently ($w_{x,y} \neq \mathbf{1}$), assigning greater or lower importance to specific areas. To address localized issues (a), we emphasize the object outline for general shape reconstruction and the hidden region in the conditioned input view for occlusion reconstruction:

$$w_{\text{outline}} = \underbrace{G_{\text{kern}} * x}_{\text{Blur}} - x \qquad \text{and} \qquad w_{\text{occluded area}} = \begin{cases} x_{\text{alpha}} \odot y_{\text{alpha}} & \text{if } x \text{ is an input view} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

The models introduced in Section 3.1 show that injecting knowledge from a large pre-trained model can be beneficial. We draw inspiration from the U-Net architecture proposed in [48]. 5×5 feature patches are extracted from DINOv2 [29], reduced from 768 to 128 channels by a convolution, and resized for concatenation with the U-Net bottleneck; cf. Figure 3. [48] also proposes a second, more powerful U-Net architecture that concatenates multi-scale features from Stable Diffusion [32] with each decoder layer. However, this has not yet been implemented in our work.

Training and Evaluation Training and evaluation are both straightforward. For each object, one rendered image is selected as the input view, while a subset of the others provides novel views from different camera angles. Splatter Image predicts 3D Gaussians, which are rendered and compared with the ground-truth renderings.
Most state-of-the-art 3D reconstruction models require substantial resources for inference and even more for training. Although Splatter Image has significantly lower resource requirements [38], its computational cost remains substantial for extensive experimentation.

The original training pipeline has two stages: initial training with only an $\mathcal{L}_2$ loss and a subsequent fine-tuning stage that also uses the LPIPS loss [49]. Because of computational constraints and to ensure a fair comparison between methods, we limit training to $25k$ iterations for most experiments, equivalent to approximately 12–24 hours on a recent single GPU. We omit the second training stage because our selected experiments show that it does not address the issues on its own.

We use three 2D metrics—LPIPS, peak signal-to-noise ratio (PSNR), and structural similarity index measure (SSIM)—for both the input (condition) view and the novel views; the latter are more important. Unfortunately, these metrics do not fully capture differences in reconstruction. For instance, a plausibly reconstructed occluded region that differs from the ground truth could receive worse scores than a blurry reconstruction.

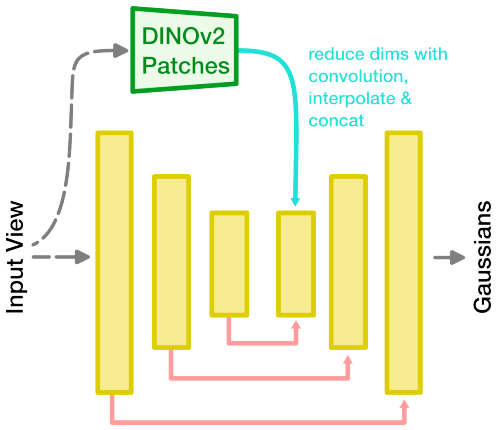


Figure 3: Simplified architecture for feature patch injection into the bottleneck of the U-Net

3.3 Experiments

In this section, we present our findings and empirical results. Training is accelerated using pretrained checkpoints from the official implementation, with weights extended to accommodate possible changes in shape. In hindsight, zero initialization would have been preferable in this context.

The core experiments and their final validation performance are presented in Table 1. Additionally, we provide example outputs in Figure 4.

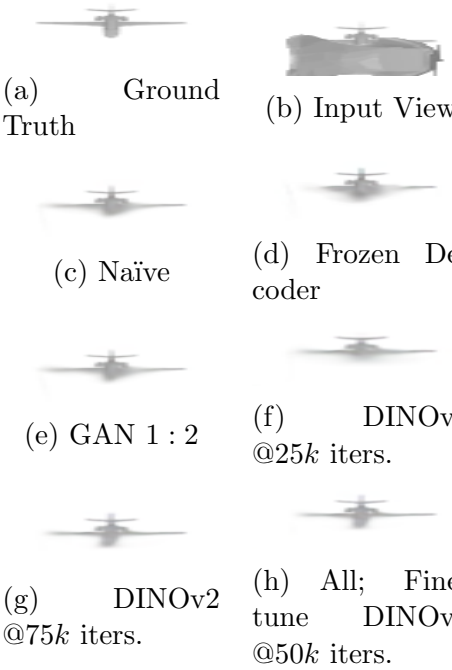
Naïve Approach We can confirm our earlier hypothesis that the original architecture of SP_0 is capable of producing rough reconstructions. Interestingly, the occlusion mask contributes only minor improvements. The model performs comparably to the SP_0 baseline on the same validation data without occlusions, particularly in terms of LPIPS and SSIM. However, the disparity is greater for PSNR, for which SP_0 significantly outperforms the model on both the conditioned and novel views; its advantage on the conditioned view is not particularly surprising.

It appears to excel at reconstructing simple geometric shapes. At the same time, we observe that the reconstructions are often blurry or lack definition and texture, particularly in regions that were occluded in the conditioned input view. This issue is common across our other models as well.

Frozen Decoder Motivated by [26], we hypothesize that a powerful model trained on standard 3D reconstruction should be adaptable to handle occlusions. Specifically, it should be sufficient for our task to learn a latent-space representation, which can then be reconstructed using the original decoder. It is important to note that U-Net [33] and its derivatives are not pure autoencoders; they include feature skip connections that directly link the encoder and decoder.

Model	PSNR↑ cond.	PSNR↑ **novel**	SSIM↑ cond.	SSIM↑ **novel**	LPIPS↓ cond.	LPIPS↓ **novel**
SP_0 without occl.	42.440,	**29.408**	0.997,	**0.954**	0.005,	**0.047**
SP_0 with occlusions*	15.938,	**15.038**	0.820,	**0.696**	0.222,	**0.295**
Naïve	32.786,	**26.674**	0.969,	**0.926**	0.050,	**0.092**
Frozen decoder	30.819,	**25.849**	0.957,	**0.916**	0.070,	**0.102**
GAN 1:2	32.837,	**26.712**	0.969,	**0.927**	0.050,	**0.091**
GAN 1:10	32.880,	**26.750**	0.970,	**0.927**	0.049,	**0.091**
GAN 1:100	31.264,	**26.092**	0.966,	**0.924**	0.059,	**0.096**
DINOv2	29.196,	**23.028**	0.950,	**0.869**	0.094,	**0.187**
DINOv2 @ 50k iters.	30.533,	**24.063**	0.960,	**0.890**	0.077,	**0.156**
DINOv2 @ 75k iters.	31.281,	**24.619**	0.964,	**0.899**	0.068,	**0.142**
Occlusion weight 2x	33.342,	**26.626**	0.972,	**0.928**	0.048,	**0.093**
Occlusion weight 5x	33.423,	**26.659**	0.972,	**0.928**	0.048,	**0.092**
Outline weight 2x	32.765,	**26.597**	0.970,	**0.927**	0.052,	**0.094**
Outline weight 5x	32.815,	**26.608**	0.970,	**0.928**	0.051,	**0.093**
All; fine-tune from SP_0	30.146,	**22.968**	0.957,	**0.869**	0.083,	**0.187**
All; fine-tune DINOv2 @ 50k iters.	31.868,	**24.579**	0.966,	**0.898**	0.063,	**0.141**

Table 1: Evaluation on the validation subset. SP_0 without occlusions is a baseline showing how the original model performs on unoccluded data. Baseline or reference experiments are shown in blue, and core experiments are shown in red. PSNR denotes peak signal-to-noise ratio, SSIM denotes structural similarity index measure, and LPIPS denotes Learned Perceptual Image Patch Similarity. "cond." denotes the conditioned input view, "occl." denotes occlusion, and "iters." denotes training iterations. *The model was not trained for this task.



(a) Ground Truth (b) Input View (c) Naïve (d) Frozen Decoder (e) GAN 1 : 2 (f) DINOv2 @25k iters. (g) DINOv2 @75k iters. (h) All; Fine-tune DINOv2 @50k iters.

Figure 4: Comparison of various models on the same conditioned input view. Results for the novel views are identical for the occlusion- and outline-weighted models.

Qualitative analysis of decoder freezing shows that the approach can yield comparable results, while exacerbating blurry regions and producing (colorful) artifacts. On average, the reconstructed shape appears worse than that produced by the naïve approach. This raises the question of whether relaxing the freezing strategy—for example, by continuing to train the skip-connection layers—could resolve the issue.

Loss Weighting The idea of *giving greater importance to the reconstruction of occluded areas via loss weighting* requires us to include the input view in the training loss, which [38] does not do. We test occlusion and outline weighting with coefficients of 2 and 5, respectively, each applied independently of the other. All four models perform similarly, with only

minor differences in their scores. On their own, the current version of loss weighting is insufficient to make a significant impact. The models perform comparably to the naïve model and are therefore among the best in this series of experiments based on PSNR, SSIM, and LPIPS.

DINOv2 Features Empirically, it seems most effective to pass the full input view, including occlusions, to DINOv2 and to use feature patches as described in Section 3.2, i.e., intermediate outputs, rather than using a single feature vector per image. In terms of metrics, the model is noticeably worse than both the naïve and loss-weighted models. We extend training to a total of $50k$ and $75k$ iterations and observe a gradual but consistent improvement in metric performance. Despite this extended training, $\text{SP}_{\text{DINOv2}}$ appears to struggle more with basic object geometry, although texture reconstruction is somewhat satisfactory. This model requires substantially more training, and it is possible that the pretrained weights cannot be utilized as effectively due to larger discrepancies in architecture. Qualitatively, the $75k$-iteration DINOv2 model begins to approach the quality of the naïve model.

GAN Loss An important component is the ratio of reconstruction to GAN loss, as detailed in Equation 1 for training the generator. The GAN loss cannot train the Splatter Image model on its own because it only encourages the output distribution to match the ground-truth distribution; it does not enforce accurate reconstruction of a specific input. We test three GAN-to-reconstruction loss-weight ratios: 1:2 (0.5), 1:10 (10%), and 1:100 (1%).
Initially, training progresses similarly across the models, but at approximately the halfway point, the $\text{SP}_{\text{GAN1:100}}$ loss begins to increase and the metrics worsen. $\text{SP}_{\text{GAN1:2}}$ achieves slightly better PSNR and SSIM scores than the naïve model, but it is not qualitatively better; it still exhibits large, blurry regions. Inspection of the discriminator loss reveals that it quickly plateaus at a low value, indicating that the GAN loss does not contribute productively.

Combined Experiment Finally, we test the simultaneous application of all previous methods: the weighted loss function (outline and occlusion weighting), GAN loss, and DINOv2 features. We use the best values from the previous individual experiments. Neither variant—initialized from SP_0 or from $\text{SP}_{\text{DINOv2@50}k\text{iters.}}$—yields satisfactory performance. Although additional training might improve the results, the computational cost of $25k$ iterations is twice that of the naïve approach.

3.4 Discussion

The experiments clearly demonstrate that the method requires substantial further development. At the same time, we observe early indications of improved object and spatial understanding.

Because the eventual goal is to process in-the-wild images, an appropriate evaluation dataset and methodology are necessary. Unlike [30] and [26], we cannot synthetically layer 2D images for quantitative evaluation because corresponding novel views do not exist. Toward this goal, we began developing a Kubric-based [14] dataset of randomly placed 3D objects. For in-the-wild images, we may also need suitable amodal completion or segmentation models.

Evidently, a naïve approach, even with additional training guidance, is insufficient to learn the complex relationship between occluded objects and their full-view 3D reconstruction. Thus far, none of the more complex modifications have been tested sufficiently to rule them out as viable solutions. We must also acknowledge potential shortcomings in our DINOv2 and GAN approaches. GANs are notoriously difficult to train [34], and the GAN experiments should have been conducted more systematically, focusing on individual components. Specifically, the simultaneous inclusion of both an RGB discriminator and a mask discriminator has proven problematic. Regarding DINOv2, [29] reports poor performance on low-resolution images, suggesting that it might be beneficial to upscale from 64×64 (technically 70×70) to improve results.

By analogy with the core principle of the alternative approaches introduced in Section 3.1, there is reason to believe that priors from powerful 2D models, such as Stable Diffusion, could be incorporated into Splatter Image to achieve satisfactory, generalizable reconstruction performance. Future efforts should focus on injecting prior knowledge, similar to the DINOv2 experiments, and on further experimentation with GANs. Additionally, higher-quality and more diverse datasets could prove fruitful.

4 Fast 3D Object Generation

Chapter authors: Michael Neumayr and Nina Kirakosyan

4.1 Introduction

In this chapter, we focus on efficient feed-forward 3D object generation. We present two 3D generator architectures that leverage and modify components from StyleGAN2 [20], the Gaussian Decoder [2], and the Hierarchical Gaussian GAN [17]. One architecture explicitly regularizes Gaussian positions using generated depth maps as geometric priors; the other uses position anchors in a hierarchy to regularize the smaller Gaussians at the next level. Our contribution is an exploration of different GAN architectures for generating centered objects.

Background In the GAN literature, the term 3D GAN is often used synonymously with 3D-aware generation, which primarily focuses on frontal views of human or animal faces. These methods learn sufficient depth and geometry to generate multi-view-consistent images of the same face from a set of look-at camera poses [4, 35, 5, 17]. Because the poses are concentrated in similar directions, these methods often employ a dedicated learnable background generator to improve the rendered images.

As discussed in Section 2, 3DGS provides a new paradigm for 3D scene generation. Optimization-based approaches conditioned on single- or multi-view scene images and/or text prompts have achieved state-of-the-art rendering quality [39, 47]. Feed-forward models for 3D object and scene generation with 3DGS are an active topic of research. Recently, several non-optimization-based approaches to 3D Gaussian scene generation have been proposed [15, 17, 43].

Project Goal. In contrast to conventional 3D-aware generation, our task considers the more challenging setting of unconditional generation of a 3D object centered in empty space. Ideally, the generated object should be renderable from all sides along a 360° trajectory.

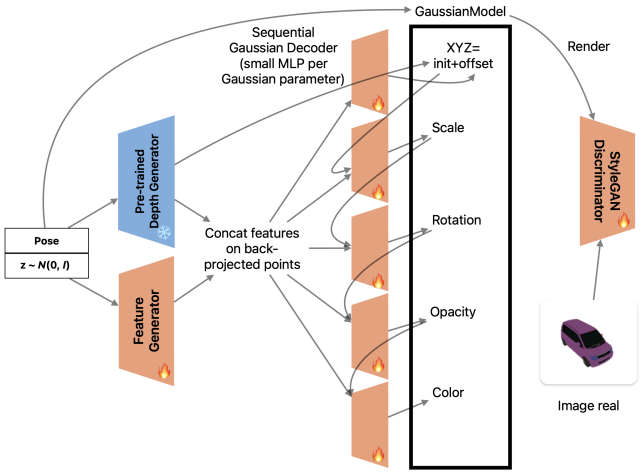
4.2 Method

The Gaussians representing an object must conform to its geometry from all viewpoints. We first test the geometric-prior setting described in Section 4.2.1. We then develop the geometry-prior-free method in Section 4.2.2, which uses hierarchical regularization to keep the Gaussians within the bounds of the object geometry.

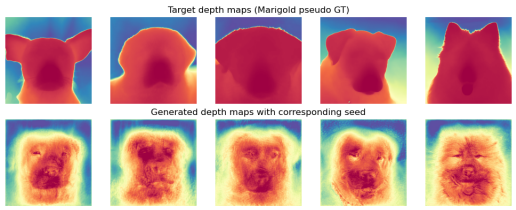
4.2.1 Depth-Generation-Based Approaches

Poor or excessively sparse initialization of Gaussian means can impede gradient flow and lead to local minima. This is particularly problematic in feed-forward architectures, which cannot rely on the non-differentiable heuristic pruning, splitting, and spawning operations used to control the number and scale of Gaussians over thousands of steps in per-scene optimization [17, 7].

Feasibility Check: Can StyleGAN2 Generate Depth? First, we validate whether a StyleGAN2 architecture can generate reasonable depth maps. Before investing resources in training a complete depth generator from scratch, we follow [3] and test whether offsets added to the style code w are sufficient. The modified code $w' = w + \delta$, where w lies in the W^+ space and $w = f(z)$, is injected into the synthesis network $g(w, n)$ to produce a depth map. Here, f and g are pretrained, frozen StyleGAN2 mapping and synthesis networks, respectively. Instead of synthesizing a natural image $x = g(w, n)$, we obtain a three-channel image $x' = g(w', n)$ and convert it into a single-channel depth map x_d by averaging its channels. We do not consider the other modalities explored in [3].
We reimplement the method from [3] and optimize the offsets on the Animal Faces-HQ (AFHQ) Dogs dataset [10]. The optimization objective uses the Huber loss, and gradient descent solves for the offsets: $\delta^* = \arg\min_\delta \mathcal{L}_{\text{Huber}}\left(P(\mathbf{x}), \mathbf{x}_d\right)$. In our case, a Marigold [21] depth-prediction pipeline $P(\mathbf{x})$ provides pseudo-ground-truth depth estimates.
Figure 5b confirms that StyleGAN2 can learn depth-related structure. On closer inspection, however, the generated depth maps are less coherent than the pseudo-ground truth and contain features that resemble edge predictions rather than depth. We therefore train a depth-specific StyleGAN2 from scratch using proper depth ground truth.



(a) Back-projection of a generated depth map. The resulting points initialize the Gaussian means before the offsets and remaining parameters are predicted sequentially.



(b) The modified code $w' = w + \delta^*$, with optimized δ^*, is passed to the pretrained StyleGAN2.

(a) Combination of reconstruction losses.

(b) Adversarial loss.

Figure 6: Gaussians initialized from a back-projected depth map. Each sample shows an example pose (left) and its corresponding rendering (right).

Initializing Gaussians from a Generated Depth Map Second, we train a dedicated single-channel depth-map generator based on the StyleGAN2 architecture. A Gaussian generator then uses this depth generator to initialize the Gaussian means. This architecture is motivated by two observations. First, in the feed-forward setting, positions cannot be corrected over thousands of optimization iterations through non-differentiable pruning and spawning heuristics; consequently, good point initialization is crucial for maintaining gradient flow and avoiding local minima [17, 7]. Second, following the Gaussian Decoder [2], features can be attached to 3D points and subsequently decoded into per-point Gaussian parameters.
The depth generator is a minimally modified StyleGAN2 architecture [20] with a single-channel output, trained on depth renderings from the ShapeNet v2 car category [6]. In the 3D Gaussian generator, the depth generator is frozen and produces depth maps that are back-projected into the scene to initialize the 3D Gaussian point cloud. During back-projection, the point cloud is filtered to separate the object from the background. Because the generator should produce a single centered object, the depth-map background is discarded. The feature-map generator applies the same mask and attaches features only to foreground points. A sequential Gaussian-parameter decoder based on [2] then converts the per-point features into Gaussian parameters. Only diffuse color is used; that is, the maximum SH degree is 0.
Figure 6 demonstrates that, when overfitting a single scene with reconstruction losses, the generated 3D Gaussians remain within the car's shape. The pipeline with adversarial loss, however, struggles to recover the correct geometry even in the single-scene experiment. The colors are inaccurate in both cases, although the reconstruction pipeline places no Gaussians in white regions of the car, allowing the fixed white background to remain visible.

4.2.2 Hierarchical Gaussians Pipeline

Given the shortcomings of the methods discussed above, we reimplemented and adapted a recently proposed method for feed-forward Gaussian scene generation: Adversarial Generation of Hierarchical Gaussians [17].
The hierarchical generation pipeline introduces a hierarchy of Gaussian parameters, expanding the set of Gaussians from one level to the next. This approach was reported to be fast and capable of generating adequate 3D-aware face reconstructions. Instead of naively initializing a latent code z and producing a fixed number of Gaussians, the hierarchical setup defines interdependent levels that represent the Gaussians from coarse to fine. At each level, it estimates a set of Gaussians by applying learned parameter offsets to fixed transformations of the preceding level's Gaussians.

Architecture. Figure 7 illustrates the modified generator architecture based on [17]. The generator consists of generator blocks (see Figure 8). As in the original pipeline, the generator takes a latent code $z \sim \mathcal{N}(0, I)$ and learnable initial Gaussian parameters const $\in \mathbb{R}^{N\times 3}$ as input. In addition, we supply pose information so that the model can learn pose-conditioned representations. Each layer_k takes the preceding layer's features x_{k-1} and its level-specific input and produces features x_k. The `toGaus` module transforms these features into Gaussian parameters, which `densify` combines with the parameters

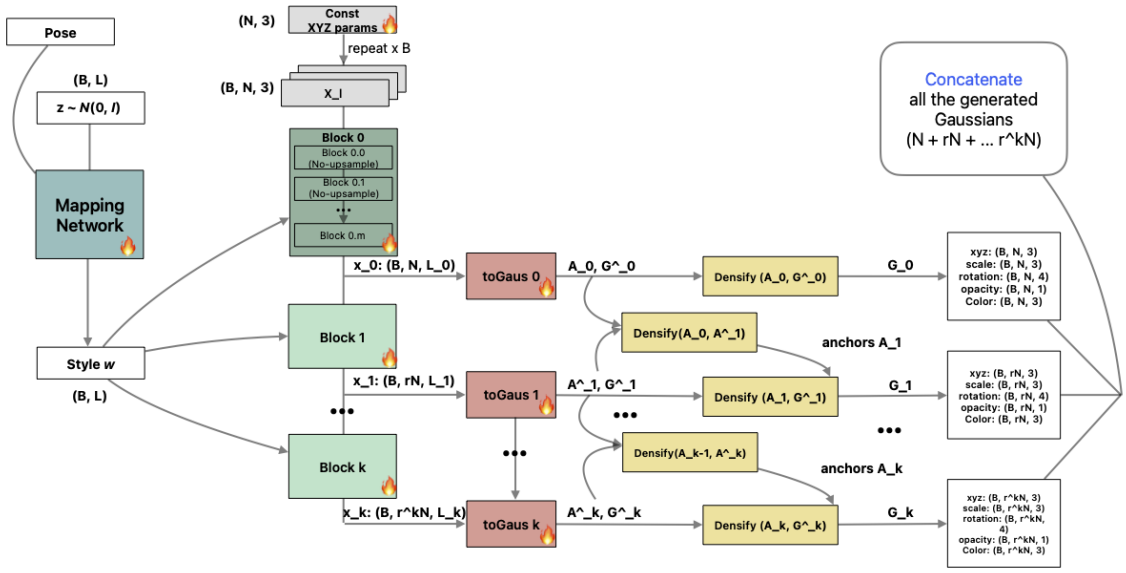


Figure 7: Hierarchical Gaussian generator architecture. Here N denotes the number of Gaussians, B denotes the batch size, L denotes the dimensionality of the latent codes z and w, L_k denotes the number of channels at level k, x_k denotes the high-dimensional features output at level k, G denotes the Gaussian parameters, $\hat{G}$ denotes the Gaussian residuals, and A denotes the Gaussian anchors.

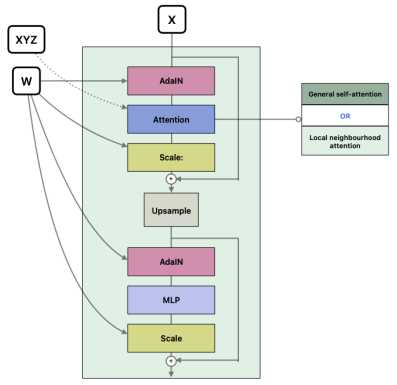


Figure 8: Block architecture.

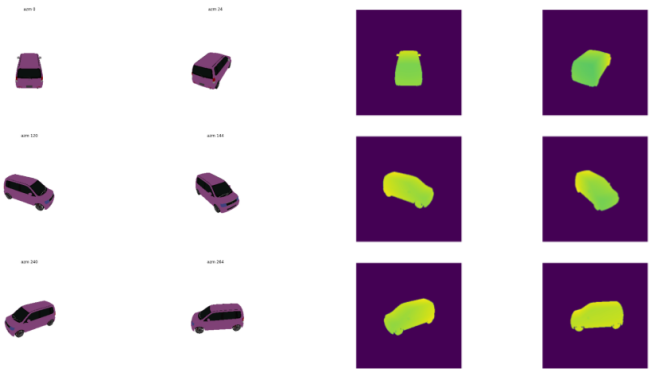


Figure 9: Custom dataset of ShapeNet v2 cars with rendered multi-view images and depth maps.

from layer$_{k-1}$. At every hierarchical level except the first, the number of Gaussians is increased by the upsampling ratio r. The final image is rendered from the combined Gaussians at all levels: $N + rN + \ldots + r^{L-1}$, where L is the number of hierarchical levels. Further architectural details are as follows.

- We use the StyleGAN2 mapping network [20] to transform the latent code z into a style code w, which is used in the generator blocks.
- The initial Gaussians const are processed by a stack of non-upsampling blocks at

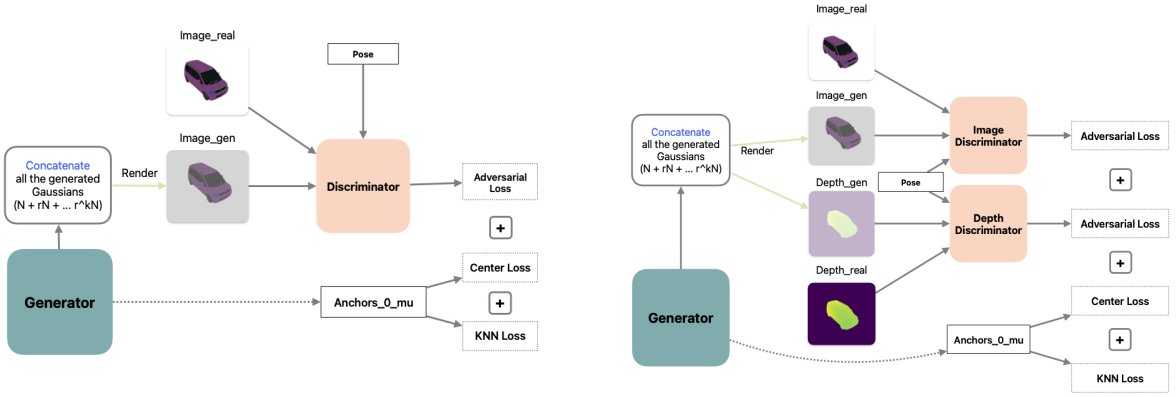
the first level level_0.

- Each generator block consists of attention, upsampling, and MLP layers with normalization and scaling. At coarser levels, we apply global self-attention across all Gaussians. At finer levels, where the number of Gaussians is larger, we use k-nearest-neighbor (k-NN) clustered local-neighborhood attention, as proposed by Zhao et al. [50].
- AdaIN and adaptive layer scaling are conditioned on the style code w [31].
- The upsampling operation is adapted from the sub-pixel operation in [36], using dense layers instead of the original convolutions on 2D data.

Hierarchical Gaussian Parameters. To transform the high-dimensional features x_k into Gaussian parameters, the features are passed through the `toGaus` module, which consists of five linear layers and Gaussian-parameter activations following the 3DGS setup [22]. This module outputs two sets of $\{\hat{\mu}, \hat{s}, \hat{\alpha}, \hat{q}, \hat{c}\}$, representing position, scale, opacity, rotation, and color estimates for the Gaussian-parameter residuals and anchors at the given level. In `densify`, the residuals are combined with the preceding level's anchor parameters to obtain the final Gaussian parameters for each level. The operations that combine parameters from adjacent hierarchical levels for level l are defined as follows.

- Position: $\mu^l = \mu^{l-1} + R^{l-1}S^{l-1}\hat{\mu}^l$, where R^{l-1}, S^{l-1} are rotation and scale matrices,
- Scale: $s^l = s^{l-1} + \hat{s}^l + \Delta s, \quad \text{where} \quad \hat{s}^l, \Delta s < 0,$
- Opacity, rotation, color: $\alpha^l = \alpha^{l-1} + \hat{\alpha}^l, \quad q^l = q^{l-1} + \hat{q}^l, \quad c^l = c^{l-1} + \hat{c}^l$

We refer interested readers to the original paper [17] for further implementation details.



(a) Training objectives for the original Adversarial Generation of Hierarchical Gaussians model.

(b) Our modified pipeline with an additional depth-discriminator objective.

Figure 10: End-to-end hierarchical Gaussian generation pipelines and their training objectives.

Training Objectives The training loss follows [17]. The adversarial loss is a non-saturating loss with R1 regularization of strength λ:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{z\sim P_z,\theta\sim P_\theta}\left[f(d(g(z,\theta)))\right] + \mathbb{E}_{I_r\sim P_{I_r}}\left[f(-d(I_r)) + \lambda\|\nabla d(I_r)\|^2\right], \tag{3}$$

where $f(t)$ is the softplus function. As in [17], we add $\mathcal{L}_{\text{center}}$ to regularize Gaussian positions and $\mathcal{L}_{\text{knn}}$ to keep distances between neighboring Gaussians small. We also experiment with an optional pose branch in the discriminator and its corresponding loss from [17], $\lambda_{\text{pose}}\mathcal{L}_{\text{pose}}$. The final weighted loss is $\mathcal{L} = \mathcal{L}_{\text{adv}} + \lambda_{\text{center}}\mathcal{L}_{\text{center}} + \lambda_{\text{knn}}\mathcal{L}_{\text{knn}}$.

Depth Discriminator. As a modification of the original pipeline, we condition the hierarchical Gaussian pipeline on ground-truth depth maps. We introduce a separate depth discriminator with the same architecture and adversarial-loss objective described above.
Figure 10 illustrates both the original end-to-end pipeline in Figure 10a and our custom setup with a depth discriminator in Figure 10b. The discriminator architectures are the same as in StyleGAN2 [20].

4.3 Experiments

4.3.1 Custom Dataset

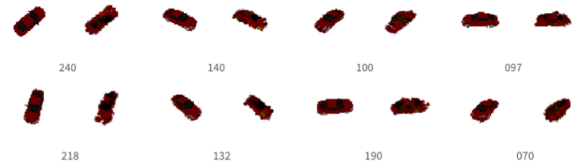
Depth-based generation approaches require a dataset of images of single-object scenes, camera poses, and ground-truth depth values. To experiment with and troubleshoot different pipeline configurations, we needed multi-view images of objects from a simple category rendered from different viewpoints.
We used objects from the "car" category of the ShapeNet v2 dataset [6]. Although many methods based on ShapeNet cars provide data and rendering implementations [37, 5], their datasets do not include corresponding depth renderings. We therefore rendered a custom car dataset using the setup proposed for Scene Representation Networks (SRNs) [37].
The "car" category contains $2.5k$ instances. We rendered 250 views of each car using the PyTorch3D library, with the car centered at the origin and the camera poses distributed on a sphere along an Archimedean spiral. The images have a resolution of 64×64. Figure 9 shows examples of RGB images and depth maps from the dataset.

4.3.2 Results

We trained our hierarchical Gaussian model, with and without a depth discriminator, on both a single scene and multiple scenes to evaluate our approach. Figure 11 illustrates selected results. Figures 11a and 11b show results obtained with different hyperparameters while overfitting the hierarchical generator to a single scene. Figure 11a uses the basic setup from the paper, whereas Figure 11b shows results from the pipeline with a depth discriminator (see Figure 10b). Figure 11c shows results from training on 30 car scenes, each with 250 poses. We generate a single asset containing the Gaussians and render it in 150–250 ms.

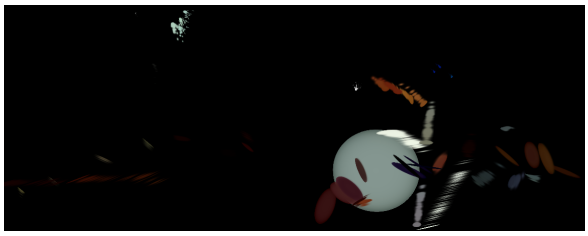
Across the tested hyperparameter settings, the model produced either highly anisotropic Gaussians scattered without a coherent form or a large cluster of Gaussians rendered as colored Gaussian noise. We hypothesize that the architecture may not yet be capable of generating dense assets with accurate geometry. Future work should investigate higher resolutions, more data, and longer training. The project timeframe did not permit the longer experimental cycles required.

(a) Adversarial loss with basic setup.



(b) Adversarial loss with depth discriminator.

(c) Adversarial loss with depth discriminator (generator trained on 30 scenes).



(d) The Gaussians generated for the object shown in Figure 11c, viewed from a different position in 3D space.

Figure 11: Overfitting to a single scene (top) and training on 30 scenes (bottom). Each sample shows example poses (left) and the corresponding renderings (right). Best viewed at high magnification.

4.4 Discussion

As illustrated, our feed-forward approaches can generate single-object scenes of cars. The models learn pose-conditioned car representations. For depth back-projection, pose conditioning is required because the 2D depth map must represent depth from the specified viewpoint, as described in Section 4.2.1. For hierarchical Gaussians, the pose-conditioned generator (see Figure 11) produces 3D Gaussians that resemble the target car when rasterized from the conditioned viewpoint, particularly when overfitting a single scene. The resemblance is weaker after training on multiple scenes. When the Gaussians are inspected from a different viewpoint, as shown in Figure 11d, their disorganized structure becomes apparent. Thus, the underlying 3D asset does not faithfully represent the car's geometry. Ideally, the feed-forward generator would produce a dense, pose-independent set of Gaussians that matches the car's geometry. This remains difficult without a geometric prior, even with a depth discriminator, and is therefore left for future work.
We also attempted a quantitative evaluation using unconditional-generation metrics. However, the Fréchet Inception Distance (FID) scores ranged from 500 to 600. We attribute this to several factors. As the examples show, the generated 3D assets have limited diversity and distinctness compared with the evaluation set, which contains dozens of distinct

cars rendered from 250 viewpoints each. This creates a substantial distribution mismatch. The limited number of generated samples and the low resolution may also affect the metric.

Unconditional object generation remains an active research frontier; reference pipelines for feed-forward generation, such as [17, 15], were first released in June 2024. Our experiments explore these recent developments.

5 Object-Centric Sparse-View 3D Gaussian Reconstruction and Decomposition

Chapter author: Wenbo Ji

5.1 Introduction

5.1.1 Motivation and Research Context

Reconstructing a scene from only a few calibrated images requires more than reproducing visible colors. A useful representation must recover geometry, appearance, and object structure. With sparse-view observations, all three are uncertain: occluded surfaces lack evidence, repeated textures weaken correspondence, and an image can look plausible despite inconsistent 3D geometry. The central problem is therefore to infer not only a renderable scene, but also a representation whose elements can be interpreted and selected.
NeRFs [27] synthesize high-quality novel views but conventionally optimize a separate representation for each scene and repeatedly sample camera rays. Sparse-view and generalizable variants reduce ambiguity through regularization, learned priors, or transferred multi-view features [28, 40, 8, 19, 42]. 3DGS [22], illustrated in Figure 1, instead represents a scene with explicit anisotropic Gaussians and renders them by rasterization. Its accessible position, opacity, covariance, and appearance parameters make it attractive for structured scene modeling, although the original method assumes dense observations and per-scene optimization.
Feed-forward methods predict Gaussians directly from sparse images. Splatter Image [38] regresses pixel-aligned Gaussians from one object-centered image, but monocular depth is ambiguous. pixelSplat [7] reconstructs general scenes from two views using epipolar features and a learned depth distribution. MVSplat [9] makes geometry more explicit through plane-sweep cost volumes. These methods emphasize appearance and geometry, whereas semantic 3DGS methods such as Gaussian Grouping [46] attach identity features for object selection but rely on dense views and scene-specific optimization. A gap remains between fast sparse-view reconstruction and object-centric understanding.
We study that gap with a feed-forward model that reconstructs and decomposes an indoor scene from two calibrated RGB images. Our hypothesis has two parts: a multi-view depth prior anchors Gaussian centers to correspondence evidence, and joint semantic prediction makes the same Gaussians directly decomposable. The geometry branch determines where evidence enters 3D space; the semantic branch determines how those primitives can be grouped. Both are supervised through rendered target views, so reconstruction and decomposition share one spatial representation.

5.1.2 Problem Definition

Given two calibrated views, we seek a mapping from the input images to a set of Gaussians that encode a center, opacity, covariance, color, and semantic feature. Rasterizing the complete set should reproduce target RGB and semantic images; selecting Gaussians by class should yield a meaningful subscene without fitting another model. Novel-view

synthesis tests appearance, depth-conditioned placement tests cross-view geometry, and semantic selection tests whether the representation contains usable object structure.

5.1.3 Contributions

The work makes four contributions:

- We construct a synthetic indoor pipeline from 3D-FRONT and 3D-FUTURE with aligned RGB, depth, semantic, instance, and camera supervision.
- We implement a two-view architecture combining transformer feature exchange, cost-volume depth estimation, and pixel-aligned Gaussian prediction.
- We jointly predict semantic Gaussian features for class-conditioned subscene selection.
- We analyze qualitative reconstruction, semantics, decomposition, and characteristic representation failures.

5.1.4 Chapter Organization

Figure 12 summarizes the chapter. Section 5.2 presents the data pipeline, Section 5.3 presents the model, and Section 5.4 presents the three evaluation questions. Section 5.5 interprets the observed limitations and future directions.

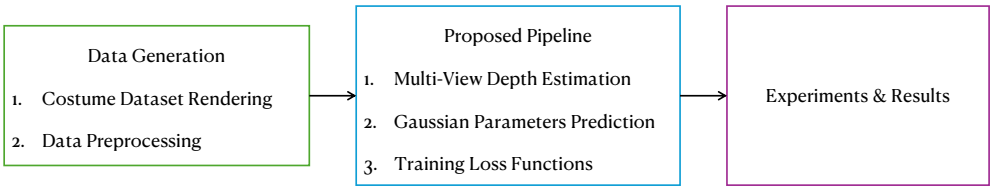


Figure 12: Overview of the project pipeline and chapter structure.

5.2 Data Generation

5.2.1 Why Synthetic Indoor Supervision

Joint geometry, appearance, and semantic learning requires pixel-aligned images, labels, depth, and cameras. Real indoor datasets often contain missing depth, annotation noise, or registration errors. Synthetic rendering provides exact calibration and consistent modalities, allowing observed errors to be attributed more directly to the representation. We use 3D-FRONT [11] for professionally designed indoor layouts and 3D-FUTURE [12] for their textured furniture models. Together they provide controlled variation in layout, viewpoint, object placement, and appearance.

5.2.2 Rendering Pipeline

The source collection contains approximately $6k$ scenes with 2–3 rooms each. We process it room by room in Blender.

Scene and label selection We retain living rooms, dining rooms, and bedrooms, which contain recurring furniture under varied arrangements and occlusions. Source categories are mapped to six labels: void, chair, table, bed, stand, and sofa. Void includes background surfaces and categories outside this vocabulary.

Camera and modality generation We sample valid camera poses inside each room and orient them toward the interior. The views must share enough content for correspondence while retaining sufficient parallax for depth estimation. For each room, we render 60 calibrated keyframes. At every pose, we render aligned RGB, depth, semantic-label, and instance-label images (Figure 13). RGB supervises appearance, semantic labels supervise Gaussian classes, and depth and instances provide geometric and object-level information for inspection.

Figure 13: Aligned synthetic supervision: RGB, semantics, depth, and instances.

Filtering and tensor organization We remove scenes with strongly occluded views, low average image brightness, or fewer objects than the specified threshold. These filters reject blocked cameras, poorly illuminated interiors, and trivial decomposition cases. Approximately $2k$ scenes remain. Each scene tensor stores its identifier, calibrated cameras, RGB images, depth maps, and semantic labels with synchronized view indices. During training, two source views and a target view are selected from the same room.

5.2.3 Dataset Limitations

The six-class vocabulary is coarse: it merges unmodeled content into void and cannot distinguish instances of the same class. Synthetic textures, lighting, and exact cameras also omit sensor noise, exposure variation, calibration error, and material complexity. Moreover, the camera generator and scene layouts induce a narrower occlusion distribution than unconstrained real observations. Restricting the data to three room types and filtering difficult views further reduces diversity. The dataset therefore supports controlled representation learning, but not a claim of real-world generalization; richer semantics, difficult occlusions, and synthetic-to-real transfer remain open problems.

5.3 Method

5.3.1 Architecture Overview

We begin with K sparse-view images $\mathcal{I} = \{I^i\}_{i=1}^K$, where $I^i \in \mathbb{R}^{H\times W\times 3}$, and camera matrices $P = \{P^i\}_{i=1}^K$. Each $P^i = K^i[R^i \mid t^i]$ contains intrinsics, rotation, and translation. The formulation supports multiple views; our experiments use two. We learn

$$f_\theta : \{(I^i, P^i)\}_{i=1}^K \mapsto \{(\mu_j, \alpha_j, \Sigma_j, c_j, s_j)\}_{j=1}^{H\times W\times K}, \tag{4}$$

where each input pixel predicts one Gaussian center μ_j, opacity α_j, covariance Σ_j, color c_j, and semantic feature s_j.
Figure 14 summarizes the tensor flow. A shared encoder and transformer exchange information across views. Plane sweeping converts these features into depth costs, and a 2D U-Net [33] predicts per-view depth maps. Unprojection determines the Gaussian centers; a parallel head predicts all remaining attributes. The same Gaussians are rasterized into target RGB and semantic outputs. Geometry is thus anchored to correspondence, while appearance and semantics remain attached to the resulting spatial primitives.

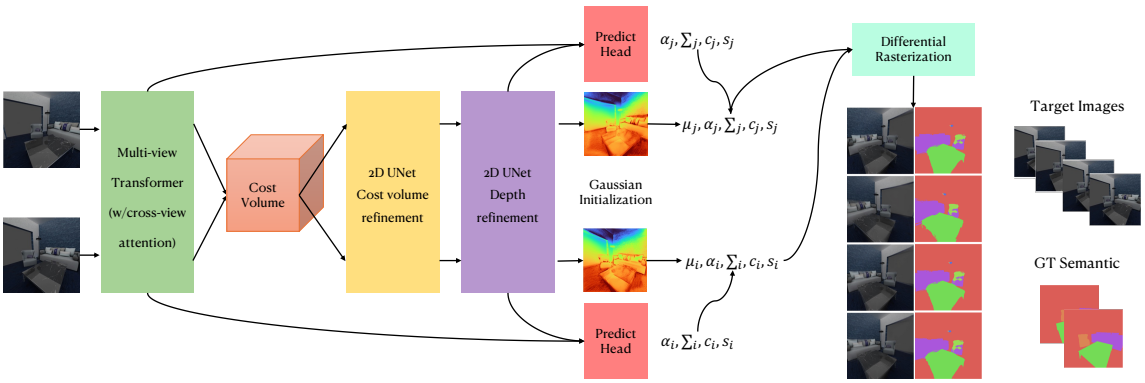


Figure 14: The proposed model predicts a depth-conditioned Gaussian representation with jointly rendered appearance and semantics.

5.3.2 Multi-View Feature Extraction

Each image is encoded as $F^i \in \mathbb{R}^{H/4\times W/4\times C}$. Self-attention adds within-view context, while cross-attention exchanges evidence between views. The resulting features remain pixel-aligned with the camera rays but can disambiguate a local patch using the second image. These features are used both for geometric matching and by the Gaussian-parameter head, allowing semantic prediction to benefit from multi-view context.

5.3.3 Cost-Volume Depth Estimation

Following MVSplat [9], we explicitly test correspondence with a plane-sweep cost volume [44, 45]. Given near and far bounds, D depths $\{d_m\}_{m=1}^D$ are sampled uniformly in inverse depth. For reference view i, features from view j are warped at every hypothesis:

$$F_{d_m}^{j\to i} = \mathcal{W}(F^j, P^i, P^j, d_m) \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times C}, \quad m = 1, 2, \ldots, D. \tag{5}$$

Their normalized dot product with the reference features produces

$$C^i_{d_m} = \frac{F^i \cdot F^{j\to i}_{d_m}}{\sqrt{C}} \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}}, \quad m = 1, 2, \ldots, D. \tag{6}$$

Stacking the correlations yields

$$C^i = [C^i_{d_1}, C^i_{d_2}, \ldots, C^i_{d_D}] \in \mathbb{R}^{\frac{H}{4}\times\frac{W}{4}\times D}. \tag{7}$$

A 2D U-Net refines C^i together with the transformer features. The expected depth under the softmax distribution is

$$V^i = \mathrm{softmax}(\hat{C}^i)[d_1, d_2, \ldots, d_D]^\mathsf{T} \in \mathbb{R}^{H\times W}. \tag{8}$$

A second 2D U-Net module refines V^i at image resolution. This reduces unconstrained 3D placement to a depth decision along a calibrated ray, although weak texture and occlusion can still make the cost distribution ambiguous.

5.3.4 Gaussian Parameter Prediction

Centers μ We unproject each refined depth map into a common world coordinate system. The deterministic union of the per-view point clouds defines the centers. It retains both views but does not merge duplicate observations of the same surface.

Opacity α The maximum of $\mathrm{softmax}(\hat{C}^i)$ provides matching confidence. Two convolutional layers convert it into opacity, allowing uncertain correspondence to contribute less strongly.

Covariance Σ and color c Two convolutional layers process the image features, refined cost volume, and RGB input. Following feed-forward 3DGS methods [7, 38], covariance is parameterized by scaling and quaternion rotation. Color is evaluated from predicted spherical-harmonic coefficients.

5.3.5 Semantic Decomposition

The same head predicts a six-class feature s_j for every Gaussian. Because s_j shares the center, covariance, opacity, and rasterization path of c_j, target-view semantic supervision constrains the reconstructed primitives rather than a separate 2D decoder. Let $\hat{y}_j$ denote the assigned class. A requested class q defines

$$\mathcal{G}_q = \{(\mu_j, \alpha_j, \Sigma_j, c_j, s_j) \mid \hat{y}_j = q\}. \tag{9}$$

Rasterizing $\mathcal{G}_q$ yields a class-conditioned subscene without retraining. The current vocabulary selects a class rather than a specific instance.

5.3.6 Training Objectives

We supervise rendered RGB with an $\mathcal{L}_2$ term and LPIPS [49], and rendered semantics with a cross-entropy loss:

$$\mathcal{L} = \mathcal{L}_2 + 0.05\mathcal{L}_{\text{LPIPS}} + \mathcal{L}_{\text{sem}}. \tag{10}$$

All losses act through the same differentiable rasterizer. Consequently, incorrect placement can affect both color and class prediction, which couples reconstruction quality to decomposition quality.

5.4 Experiment

5.4.1 Experimental Protocol

Dataset and implementation We use approximately $2k$ filtered scenes from Section 5.2.2 with an 80:20 training–test split. The model is implemented in PyTorch with a custom CUDA Gaussian rasterizer. The transformer architecture and depth-sampling strategy follow MVSplat [9]. We train for 250,000 iterations on a GPU with 48 GB of memory using Adam [23].

Evaluation scope Because the current evidence is qualitative, we do not claim numerical superiority or generalization beyond the synthetic test setting. We ask three research questions (RQs):

- **RQ1:** Can two input views support coherent novel-view reconstruction?
- **RQ2:** Are the rendered semantic predictions spatially consistent?
- **RQ3:** Do semantic Gaussian features support useful decomposition?

5.4.2 RQ1: Reconstruction Quality

Figure 15 shows that the model preserves the dominant room layout, central table, and left cabinet across nearby target viewpoints. Large wall, floor, and tabletop regions remain coherent, indicating that the cost volume establishes useful correspondence from two inputs.

The central table nevertheless has softened or doubled contours, and the cabinet boundary shows blur against the wall. The likely cause is a small disagreement between per-view depths: direct point-cloud union places redundant Gaussians near, rather than on, the same surface. The depth maps recover the broad near-to-far layout and a distinct foreground table, but their silhouettes are less stable than large planar interiors. Occlusion and weak texture reduce matching evidence, and RGB blending can conceal geometric inconsistency that remains visible in depth.

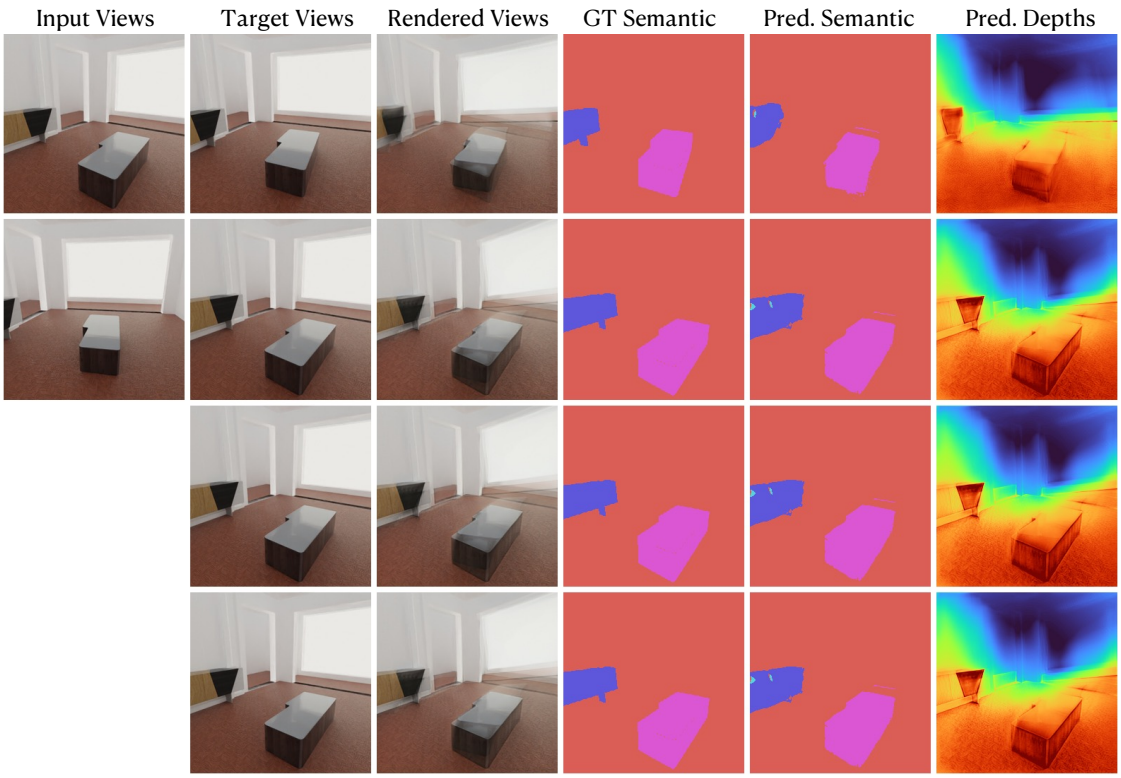


Figure 15: Input views, target views, rendered RGB, ground-truth and predicted semantics, and predicted depths.

5.4.3 RQ2: Semantic Prediction Quality

Semantic predictions are stable over dominant region interiors: the background and central table retain consistent class colors across target views, and the left furniture is recognized at a coarse scale. Since these labels are rendered from Gaussian features, the agreement suggests that semantics are carried by the 3D primitives rather than independently decoded per target image.
Errors concentrate at contours and small regions. Thin incorrect strips appear near the left furniture, and the table silhouette deviates from the target. These failures combine classification ambiguity with geometry: a misplaced or oversized Gaussian renders its semantic feature into the wrong region. Stable decomposition therefore requires both accurate features and boundary-aware placement.

5.4.4 RQ3: Object-Centric Decomposition

Figure 16 renders semantic subsets separately. Dominant furniture remains recognizable after other classes are removed, and the subsets retain appearance, showing that geometry, color, and semantics share the same primitives. Thin, distant, or occluded objects are more fragmented; neighboring residuals remain near some boundaries. Moreover, class selection cannot distinguish two instances with the same label. The result demonstrates object-level accessibility, but not yet instance-consistent decomposition.

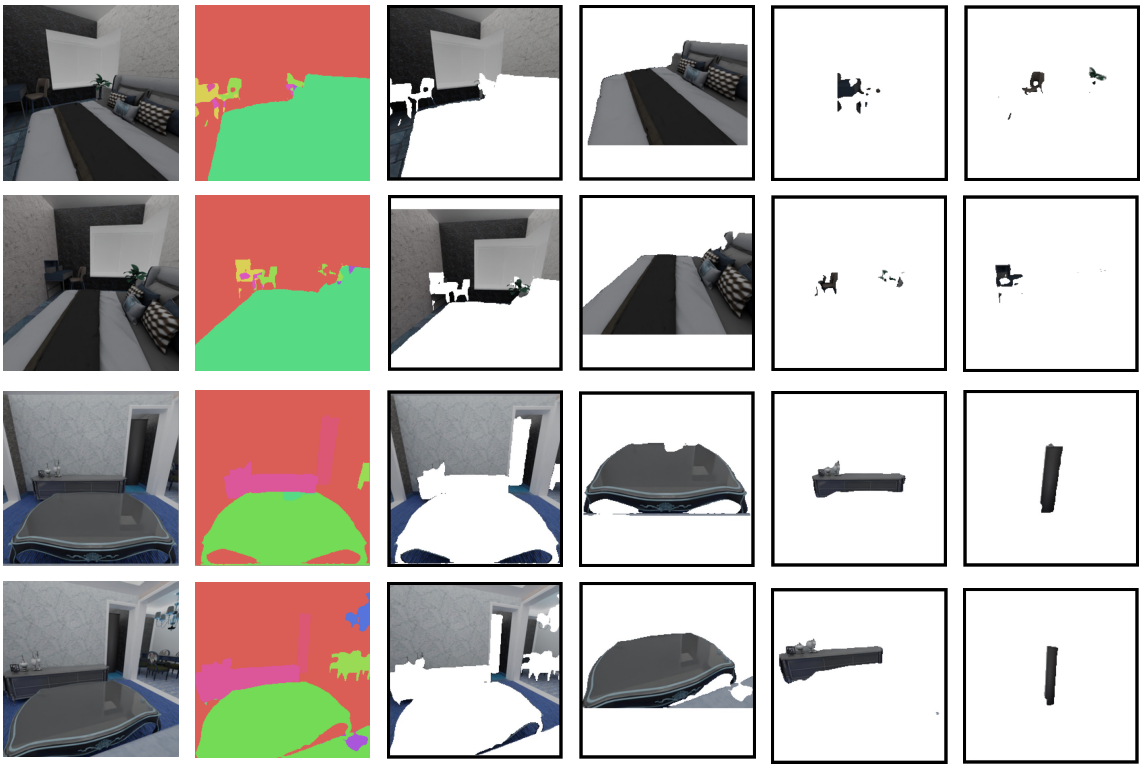


Figure 16: Class-conditioned Gaussian subsets rendered as separate subscenes.

5.4.5 Experimental Finding

The three questions expose a shared failure chain: uncertain depth creates redundant or misplaced Gaussians; these blur RGB boundaries and carry semantic features into neighboring regions; selection then reveals the inconsistency. The appropriate target for improvement is therefore the representation and its fusion mechanism, not only the final image decoder.

5.5 Discussion

5.5.1 Findings and Limitations

The project shows that a feed-forward model can turn two calibrated images into a representation that supports novel-view rendering, semantic prediction, and class-conditioned selection. Cost-volume depth recovers the dominant layout, and semantic features remain stable over major furniture regions. More importantly, the failures identify four coupled limitations.

Sparse-view depth uncertainty Plane sweeping constrains geometry along camera rays, but textureless areas, repeated patterns, narrow baselines, and occlusions can produce an ambiguous depth distribution. An incorrect center then affects every attached attribute, appearing as RGB blur, a semantic error, and an incomplete selected object.

Cross-view redundancy Deterministically uniting dense per-view point clouds does not determine whether two pixels observe the same surface. Small depth discrepancies create duplicate Gaussians with different color, opacity, or semantics. Blending may hide this redundancy in RGB while leaving the representation inconsistent.

Semantic boundaries Large interiors are more stable than contours and thin structures. A Gaussian near a depth discontinuity can cover both sides of a boundary, and the six-class vocabulary cannot distinguish instances of one category.

Synthetic-to-real gap Synthetic images provide exact cameras and labels but omit real sensor noise, calibration error, exposure changes, complex materials, and unconstrained clutter. The filtered three-room dataset also narrows viewpoint and scene diversity. The current evidence therefore concerns a controlled setting rather than real-world robustness.

5.5.2 Future Research Directions

These limitations motivate a progression of representation questions.

Correspondence-guided fusion Cross-view matching could determine which per-view Gaussians describe the same surface. A learned fusion stage could combine compatible position, appearance, and semantics while using cost-volume confidence to preserve genuinely separate or occluded observations.

Compact and consistent Gaussians A feed-forward model should predict the primitives required by a scene rather than retain one per input pixel. Visibility-aware consolidation, pruning, or consistency objectives could reduce duplication and make each Gaussian a more stable unit for editing and reasoning.

Open-vocabulary semantics Language-aligned features could replace the fixed six-class output, while instance cues could distinguish multiple objects of one category. The challenge is to preserve rich semantics across views without transferring noisy 2D boundaries into 3D.

Dynamic scenes and temporal consistency Extending the representation from static 3D to 4D requires separating camera motion, object motion, and changing visibility while preserving identity. The present depth and semantic branches provide a starting point, but temporal correspondence must keep an object addressable as it moves or becomes occluded.

Embodied perception and interaction For robots, reconstruction is useful when objects can be localized, tracked, selected, and related to free space or actions. Object-centric Gaussians could become a structured world representation connecting visual evidence to spatial reasoning and interaction. This shifts evaluation from photorealism alone toward whether geometry and semantics remain reliable for decisions.

The sequence is deliberate: sparse-view reconstruction supplies geometry, decomposition exposes components, temporal modeling tests their persistence, and embodied interaction tests their utility. More broadly, the experiments show that plausible rendered images can conceal structural inconsistency. This observation shifts the research focus from implementing an image-generation pipeline to diagnosing the correspondence, fusion, and semantic bottlenecks of the underlying representation.

References

[1] Jiayang Ao, Qiuhong Ke, and Krista A. Ehinger. "Image amodal completion: A survey". In: *Computer Vision and Image Understanding* 229 (2023), p. 103661.

[2] Florian Barthel et al. *Gaussian Splatting Decoder for 3D-aware Generative Adversarial Networks*. 2024. arXiv: `2404.10625 [cs.CV]`.

[3] Anand Bhattad et al. "Stylegan knows normal, depth, albedo, and more". In: *Advances in Neural Information Processing Systems* 36 (2024).

[4] Eric R Chan et al. "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5799–5809.

[5] Eric R Chan et al. "Efficient geometry-aware 3d generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16123–16133.

[6] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[7] David Charatan et al. "pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 19457–19467.

[8] Anpei Chen et al. "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 14124–14133.

[9] Yuedong Chen et al. "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images". In: *arXiv preprint arXiv:2403.14627* (2024).

[10] Yunjey Choi et al. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[11] Huan Fu et al. "3d-front: 3d furnished rooms with layouts and semantics". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10933–10942.

[12] Huan Fu et al. "3d-future: 3d furniture shape with texture". In: *International Journal of Computer Vision* 129 (2021), pp. 3313–3337.

[13] Jun Gao et al. "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images". In: *Advances In Neural Information Processing Systems*. 2022.

[14] Klaus Greff et al. "Kubric: a scalable dataset generator". In: (2022).

[15] Paul Henderson et al. "Sampling 3D Gaussian Scenes in Seconds with Latent Diffusion Models". In: *arXiv preprint arXiv:2406.13099* (2024).

[16] Yicong Hong et al. "Lrm: Large reconstruction model for single image to 3d". In: *arXiv preprint arXiv:2311.04400* (2023).

[17] Sangeek Hyun and Jae-Pil Heo. *Adversarial Generation of Hierarchical Gaussians for 3D Generative Model*. 2024. arXiv: 2406.02968 [cs.CV].

[18] Hanwen Jiang, Qixing Huang, and Georgios Pavlakos. "Real3D: Scaling Up Large Reconstruction Models with Real-World Images". In: (2024).

[19] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. "Geonerf: Generalizing nerf with geometry priors". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18365–18375.

[20] Tero Karras et al. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proc. CVPR*. 2020.

[21] Bingxin Ke et al. "Repurposing diffusion-based image generators for monocular depth estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9492–9502.

[22] Bernhard Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Trans. Graph.* 42.4 (July 2023).

[23] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[24] Jonas Kulhanek et al. "WildGaussians: 3D Gaussian Splatting in the Wild". In: *arXiv* (2024).

[25] Ruoshi Liu et al. *Zero-1-to-3: Zero-shot One Image to 3D Object*. 2023. arXiv: 2303.11328 [cs.CV].

[26] Zhengzhe Liu et al. "Object-level Scene Deocclusion". In: *SIGGRAPH 2024 Conference Papers* (2024).

[27] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[28] Michael Niemeyer et al. "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5480–5490.

[29] Maxime Oquab et al. "Dinov2: Learning robust visual features without supervision". In: *arXiv preprint arXiv:2304.07193* (2023).

[30] Ege Ozguroglu et al. "pix2gestalt: Amodal segmentation by synthesizing wholes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 3931–3940.

[31] William Peebles and Saining Xie. "Scalable Diffusion Models with Transformers". In: *arXiv preprint arXiv:2212.09748* (2022).

[32] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241.

[34] Divya Saxena and Jiannong Cao. “Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions”. In: *ACM Comput. Surv.* 54.3 (May 2021).

[35] Katja Schwarz et al. “Graf: Generative radiance fields for 3d-aware image synthesis”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20154–20166.

[36] Wenzhe Shi et al. “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1874–1883.

[37] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. “Scene representation networks: Continuous 3d-structure-aware neural scene representations”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[38] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. “Splatter Image: Ultra-Fast Single-View 3D Reconstruction”. In: *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[39] Jiaxiang Tang et al. “DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation”. In: *arXiv preprint arXiv:2309.16653* (2023).

[40] Prune Truong et al. “Sparf: Neural radiance fields from sparse and noisy poses. IEEE”. In: *CVF Conference on Computer Vision and Pattern Recognition, CVPR*. Vol. 2. 3. 2023, p. 6.

[41] Vikram Voleti et al. “SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion”. In: *European Conference on Computer Vision (ECCV)*. 2024.

[42] Qianqian Wang et al. “Ibrnet: Learning multi-view image-based rendering”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 4690–4699.

[43] Dejia Xu et al. “Agg: Amortized generative 3d gaussians for single image to 3d”. In: *arXiv preprint arXiv:2401.04099* (2024).

[44] Haofei Xu et al. “Unifying flow, stereo and depth estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[45] Yao Yao et al. “Mvsnet: Depth inference for unstructured multi-view stereo”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 767–783.

[46] Mingqiao Ye et al. “Gaussian grouping: Segment and edit anything in 3d scenes”. In: *arXiv preprint arXiv:2312.00732* (2023).

[47] Taoran Yi et al. “GaussianDreamer: Fast Generation from Text to 3D Gaussians by Bridging 2D and 3D Diffusion Models”. In: *CVPR*. 2024.

[48] Guanqi Zhan et al. “Amodal ground truth and completion in the wild”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 28003–28013.

[49] R. Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2018, pp. 586–595.

[50] Hengshuang Zhao et al. “Point transformer”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 16259–16268.