SplatDiffuser: Generative 3D Reconstruction from a Single Image

Robert Ren Ziwei Liao Ellina Zhang Steven Waslander
University of Toronto

Abstract

We address the ill-posed problem of single-image 3D reconstruction and generation. Existing deterministic methods regress only the mean of the scene distribution, leading to blurred geometry and poor generalization in ambiguous or unseen regions. To overcome these limitations, we propose SplatDiffuser, a generative model that performs conditional denoising diffusion directly over 3D Gaussian Splat parameters, enabling sampling of multiple plausible and view-consistent reconstructions. Unlike conventional diffusion approaches that first compress high-dimensional Gaussians into a latent space—which can introduce high-frequency loss through the VAE bottleneck and requires sub-optimal two-stage training—our framework models the distribution directly in 3D Gaussian space, a design choice that avoids potential information loss from latent compression and allows end-to-end training with stable convergence. SplatDiffuser acts as a projective denoiser built around a feed-forward Gaussian predictor with differentiable rendering as an inductive bias, enabling training purely from image supervision. Trained on CO3D (hydrants, teddybears) and RealEstate10K, our method achieves competitive reconstruction quality with inference in seconds and real-time rendering, while producing controllable, diverse, and realistic 3D scenes consistent across multiple views.

1. Introduction

3D reconstruction [3, 35] seeks to recover dense and geometrically consistent 3D representations of real-world scenes from sensor observations. Such reconstructions are fundamental for enabling machines and robots to perceive and understand spatial structures, interact with their surroundings for navigation and manipulation [2, 33], build world representations that serve as simulators for agents to learn from their environments [1, 34], and support immersive applications such as augmented reality, where virtual content must align with real-world geometry [27, 28].

Recent advances have introduced new scene representations, such as Neural Radiance Fields (NeRF) [25] and 3D Gaussian Splatting [19], which demonstrate impressive performance in modeling high-quality scene textures [19], geometry [10], and semantics [11], and in supporting novel view synthesis. However, these methods often rely on a dense set of input images to sufficiently constrain the high-dimensional parameters of the representation. Regions with sparse or missing observations tend to produce blurry reconstructions or visual artifacts. Moreover, collecting such dense multi-view data is costly and often impractical in real-world scenarios.

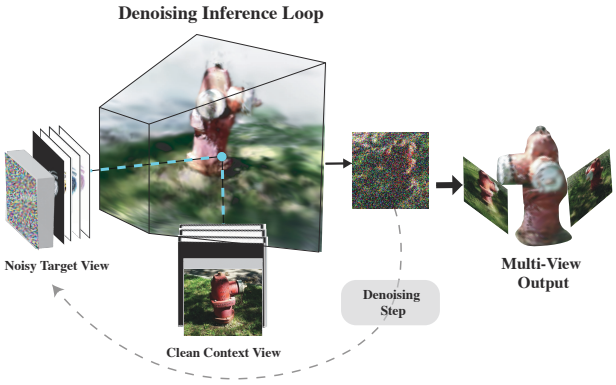


Figure 1. Starting from a noisy target view, SplatDiffuser iteratively predicts and refines a 3D Gaussian Splat representation conditioned on a clean context view. Each denoising step renders the current Gaussians to update the target estimate. The final 3D splats yield photorealistic multi-view renderings.

We aim to address 3D reconstruction from single-view images, thereby enabling broader applicability when only limited visual input is available. This is a highly ill-posed inverse problem [18], as single-view observations lead to depth ambiguity and underfitting in unobserved regions [12]. Recent methods [5, 6, 39, 40] aim to reduce the input requirements to a few views, or even a single view, by training neural networks to directly infer 3D representations from images. However, these approaches typically formulate 3D reconstruction as a *regression* problem. When only single-view images are available, the network tends to predict a single mode over all possible configurations, resulting in blurry reconstructions and artifacts in ambiguous

regions [5, 40], particularly when the camera moves into unseen views [6, 39]. In contrast, we formulate 3D reconstruction as a *conditional generation* [14, 20] problem from a single image. Rather than predicting a single most-likely output ("maximum posteriori") [2], we aim to model the entire distribution of possible 3D representations, allowing us to sample diverse and high-quality reconstructions.

To this end, we introduce a denoising diffusion model [14, 38], which has demonstrated remarkable capability in modeling complex, high-dimensional distributions across images [32], videos [15], and 3D [17, 21, 30] domains. We leverage this framework to learn the distribution of Gaussian Splatting parameters directly.

Recent works have explored integrating diffusion models with 3D Gaussian Splatting, but they differ fundamentally in the distributions they target: (1) *2D Image Distributions.* A line of work [9, 24, 30] leverages pre-trained 2D image diffusion models as priors, aggregating multi-view information through techniques such as Score Distillation Sampling (SDS) [30] to reconstruct 3D representations. However, these methods suffer from multi-view inconsistency (e.g., multi-face artifacts [30]) and require hours of costly optimization across multiple views [9]. (2) *3D Distributions of Objects.* Another line of research [44, 46, 47] focuses on directly modeling the 3D scene distribution using NeRF or Gaussian Splatting. However, diffusion training [38] requires access to ground-truth samples $X \sim P(X)$, to which noise is added, and the denoiser is trained to recover the original clean samples. As a result, these methods rely on large-scale, high-quality 3D datasets, often limited to objects such as Objaverse [7], which constrains scalability, diversity, and generalization, particularly for complex real-world scenes [48]. (3) *Latent Space Distributions of 3D Scenes.* The most related works, Bolt3D [41], Henderson et al. [13] and Liao et al. [22], employ latent diffusion [32] and train from images to model 3D scenes. However, they model a distribution over *latent variables*, which must then be decoded back into Gaussian parameters. Their reliance on a latent bottleneck (via a VAE [20]) can cause high-frequency detail loss and requires sub-optimal two-stage optimization.

In contrast to the approaches above, we propose a generative pipeline based on a denoising diffusion model that *directly models the 3D distribution of scenes* using Gaussian Splatting, without relying on any latent space, and trained solely with image-level supervision. This design enables large-scale training from widely available image datasets [31, 48]. By removing the dependency on latent representations, our approach avoids potential information loss from latent compression by design and allows stable, end-to-end optimization.

We further introduce key technical components to make diffusion training effective for Gaussian Splatting: a projective denoiser that integrates a feed-forward Gaussian predictor with differentiable rendering [19] as an inductive bias, enabling pure image-level supervision. Architecturally, we design a two-view denoiser based on DiT [29] with cross-attention to jointly process the conditioning inputs and noise, and iteratively aggregate information from both the contextual and noisy views to progressively denoise the Gaussian representations.

Our main contributions are summarized as follows:

- We propose a diffusion model over 3D Gaussian Splats that reconstructs a scene from a single image, avoiding the detail loss a latent bottleneck can introduce and enabling diverse sampling in ambiguous regions.
- We introduce an end-to-end diffusion training pipeline that learns a distribution of 3D Gaussian Splats directly from images, eliminating the need for a latent space by operating entirely in the 3D Gaussian Splatting domain.
- We achieve competitive reconstruction performance on CO3D and Re10K with computationally efficient training and inference, while also demonstrating the ability to generate diverse samples.

2. Related Work

2.1. Sparse-view Feed-forward 3D Reconstruction

Like our approach, feed-forward reconstructors predict an explicit, directly renderable 3D representation from one or a few images using a differentiable renderer, most commonly 3D Gaussian Splatting [19]. Early generalizable baselines such as pixelNeRF [45] condition a NeRF on sparse views but require volumetric rendering. In contrast, recent methods operate explicitly in Gaussian space for speed and scalability. For *objects*, Splatter Image [40] performs truly monocular reconstruction at tens of FPS by regressing per-pixel splats, while pixelSplat [5] reconstructs from pairs of images by sampling Gaussian means from a learned volumetric probability, improving robustness to sparsity. For *scenes*, Flash3D [39] anchors layered Gaussians to monocular depth to fill behind occlusions from a single image. Feed-forward *few-view* models strengthen geometry via cross-view cues: MVSplat [6] builds plane-sweep cost volumes to localize Gaussian centers; PF3plat [16] relaxes camera-pose assumptions using monocular depth and correspondence priors; large reconstruction models (GRM [44], GS-LRM [47], GeoLRM [46]) fuse sparse-view tokens with transformer decoders to emit pixel-aligned Gaussians in sub-second inference. To improve generalization and memory, latentSplat [43] autoencodes scenes into variational *latent* Gaussians decoded by a lightweight 2D generator, while PixelGaussian [8] adapts the number and distribution of Gaussians to local geometric complexity to handle arbitrary view counts. Our method builds directly on this feed-forward, Gaussian-explicit method—retaining its effi-

ciency and differentiable rasterization—while introducing a conditional diffusion formulation that yields a *distribution* over plausible reconstructions from a single image without sacrificing single-pass efficiency.

2.2. Diffusion Models for 3D Reconstruction

A parallel line of work uses diffusion to produce 3D reconstructions with a generative distribution, either by operating in image space and lifting to 3D or by denoising directly in an explicit 3D representation. DFM [42] integrates a differentiable forward model into the diffusion loop to learn distributions over *unobserved* 3D NeRFs from only image supervision; however, inference typically requires a full denoising process for each novel view. Several methods move diffusion into Gaussian Splatting (GS). GSD [26] learns an *unconditional* 3DGS prior and performs view-guided denoising for single-image object reconstruction on *CO3D* [31]. DiffusionGS [4] "bakes" differentiable splatting into the denoiser and predicts pixel-aligned GS primitives at each step, reporting fast single-stage image-to-{object,scene} 3D on *RealEstate10K*. Complementary to direct 3D denoising, latent approaches compress GS scenes and run multi-view diffusion in the latent space for sub-second synthesis and single/few-view conditioning [13]. Hybrid pipelines connect multi-view diffusion priors to GS via a regression head or via latent alignment: Bolt3D [41] generates multi-view images and regresses Splatter-Images to assemble a 3DGS scene in a feed-forward pass, while DiffSplat [23] repurposes large 2D diffusion models to directly produce GS attributes, bootstrapping pseudo-3DGS data with a lightweight reconstructor and enforcing 3D coherence via rendering losses. Zero-1-to-G [24] similarly adapts pretrained 2D diffusion to *direct* GS generation using cross-view/attribute attention, and Generative Gaussian Splatting (GGS) [36] couples a 3DGS feature field with a pretrained *video* diffusion prior for multi-view-consistent scene synthesis on *RealEstate10K* [48]. Beyond pure generation, 3D diffusion priors have been used within mapping systems from sparse views [21]. In contrast to these designs, our method performs conditional diffusion *directly* over explicit 3DGS parameters, trained with only image-space supervision like DFM [42] but producing a renderable scene in a *single* denoising loop, enabling real-time rendering and controllable sample diversity.

3. Method

3.1. Preliminaries

3D Gaussian Splatting We model a scene by a finite set of colored 3D Gaussians

$$S = \{(\sigma_i, \mu_i, \Sigma_i, c_i)\}_{i=1}^{N},$$

where $\sigma_i \in [0,1]$ an opacity, $\mu_i \in \mathbb{R}^3$ is the mean, $\Sigma_i \in \mathbb{S}^3_{++}$ the covariance, and $c_i(\nu) : \mathbb{S}^2 \to \mathbb{R}^3$ a (view-dependent) color, where $\nu \in \mathbb{S}^2$ is the camera direction. Each gaussian is modeled as:

$$g_i(x) = \exp\Big(-\tfrac{1}{2}(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\Big), 1 \le i \le N \quad (1)$$

Combined with the opacity σ_i and color $c_i(\nu)$, they define a radiance field:

$$\sigma(x) = \sum_{i=1}^{N} \sigma_i\, g_i(x),\ c(x,\nu) = \frac{\sum_i c_i(\nu)\,\sigma_i\, g_i(x)}{\sum_j \sigma_j\, g_j(x)} \quad (2)$$

where $\sigma(x)$ and $c(x,\nu)$ define the opacity and color for any point $x \in \mathbb{R}^3$.

Given a calibrated camera with ray $\gamma_{\mathbf{u}}(\tau) = x_0 + \tau\nu$ through pixel u where $\tau \in \mathbb{R}_+$, we render an image $I(\mathbf{u})$ from the radiance field using the standard emission-absorption integral

$$I(\mathbf{u}) = \int_0^\infty c(\gamma_{\mathbf{u}}(\tau),\nu)\,\sigma(\gamma_{\mathbf{u}}(\tau))\, e^{-\int_0^\tau \sigma(\gamma_{\mathbf{u}}(\mu))\,d\mu}\, d\tau \quad (3)$$

Splatter Image Given the task of single view 3D reconstruction we require a mapping from a single RGB image I to a 3D Gaussian mixture $S = \{(\sigma, \mu, \Sigma, c)\}$ that can be rendered by the 3DGS rasterizer. Following the *splatter image* parameterization, we implement this map by predicting, for every pixel $\mathbf{u} = (u, v, 1)$, a compact feature vector

$$M_{\mathbf{u}} = (\sigma_{\mathbf{u}},\ \mu_{\mathbf{u}},\ \Sigma_{\mathbf{u}},\ c_{\mathbf{u}}) \in \mathbb{R}^K$$

with a standard image-to-image network f. Collecting all per-pixel vectors yields a tensor $M \in \mathbb{R}^{H\times W\times K}$. This is then decoded into a set of Gaussians resulting in one Gaussian per pixel in I.

All Gaussians are expressed in the camera reference frame. Let the ray through pixel $\mathbf{u}$ be $x = d_{\mathbf{u}}\mathbf{u}$ and let $d_{\mathbf{u}} > 0$ denote the predicted depth along this ray. The 3D mean is specified by a ray-point plus a learned offset:

$$\mu_{\mathbf{u}} = \begin{bmatrix} u d_{\mathbf{u}} + \Delta_x(\mathbf{u}) \\ v d_{\mathbf{u}} + \Delta_y(\mathbf{u}) \\ d_{\mathbf{u}} + \Delta_z(\mathbf{u}) \end{bmatrix}, \quad \delta = (\Delta_x, \Delta_y, \Delta_z) \quad (4)$$

Diffusion Models Let $x_0 \in \mathbb{R}^{H\times W\times 3}$ be a clean image. Fix a schedule $\{\beta_t\}_{t=1}^T \subset (0,1)$ and define $\alpha_t := 1 - \beta_t$ and $\bar\alpha_t := \prod_{s=1}^t \alpha_s$. The forward noising process is

$$\begin{aligned} q(x_t \mid x_{t-1}) &= \mathcal{N}\big(\sqrt{\alpha_t}\, x_{t-1},\ \beta_t \mathbf{I}\big), \\ q(x_t \mid x_0) &= \mathcal{N}\big(\sqrt{\bar\alpha_t}\, x_0,\ (1-\bar\alpha_t)\mathbf{I}\big). \end{aligned} \quad (5)$$

A denoiser p_θ models the reverse conditionals, conditioned on some side information c (e.g., context images / poses):

$$\begin{aligned} p_\theta(x_{t-1} \mid x_t, c) &= \mathcal{N}\big(\mu_\theta(x_t, t, c),\ \tilde\beta_t \mathbf{I}\big), \\ \tilde\beta_t := \frac{1-\bar\alpha_{t-1}}{1-\bar\alpha_t}\,\beta_t, &\quad \alpha_t = \frac{\bar\alpha_t}{\bar\alpha_{t-1}},\ \beta_t = 1-\alpha_t. \end{aligned} \quad (6)$$

We use the x_0–parameterization: the network predicts a clean estimate $\hat{x}_{0|t} = \hat{x}_{0|t}(x_t, t, c)$, yielding the DDPM mean

$$\mu_\theta(x_t, t, c) = \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, \hat{x}_{0|t}. \quad (7)$$

The sampling step is then

$$x_{t-1} = \mu_\theta(x_t, t, c) + \sqrt{\tilde{\beta}_t}\,\epsilon_{t-1}, \qquad \epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I}), \quad (8)$$

Equivalently, one may predict noise $\hat{\epsilon}_\theta$ and set $\hat{x}_{0|t} = (x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}_\theta)/\sqrt{\bar{\alpha}_t}$.

3.2. Integrating Feed Forward Gaussian Splatting into Diffusion

Since we never directly observe the latent 3D scene, S, we must rely upon observations to recover the underlying signal. In our case, these observations are images that are produced by a known differentiable renderer $\mathcal{R} : S \times \Pi \to \mathbb{R}^{H\times W\times 3}$ and our latent scene is a set of N Gaussian splats $S = \{(\sigma_i, \mu_i, \Sigma_i, c_i)\}_{i=1}^N$.

Conditional Diffusion over Observations. Using a similar idea to [42], we integrate a feed-forward Gaussian Splatting model in the conditional denoising loop. Let I^{ctx} be a clean context image with pose Π^{ctx} and I_0^{tgt} a clean target image with pose Π^{tgt}. We diffuse only the target to obtain $I_t^{\text{tgt}} \sim q(\cdot \mid I_0^{\text{tgt}})$, the noisy target. At each reverse step we predict the latent scene, S, from the noisy target and clean context, then render a clean target estimate:

$$S_t = \mathcal{G}_\theta\big(I_t^{\text{tgt}},\ I^{\text{ctx}},\ t,\ \Pi^{\text{tgt}},\ \Pi^{\text{ctx}}\big), \quad (9)$$

$$\widehat{I}_{0|t}^{\text{tgt}} = \mathcal{R}\big(S_t;\, \Pi^{\text{tgt}}\big)\,. \quad (10)$$

We set the conditioning variable to $c = (I^{\text{ctx}}, \Pi^{\text{tgt}}, \Pi^{\text{ctx}})$ and apply the DDPM update (8) with $x_t = I_t^{\text{tgt}}$ and $\hat{x}_{0|t} = \widehat{I}_{0|t}^{\text{tgt}}$:

$$I_{t-1}^{\text{tgt}} = a_t\, I_t^{\text{tgt}} + b_t\, \widehat{I}_{0|t}^{\text{tgt}} + \sqrt{\tilde{\beta}_t}\,\epsilon_{t-1}, \quad (11)$$

where $a_t = \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$, $\quad b_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}$, and $\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$. In essence, we define a conditional generative model over the latent scene trajectory induced by the reverse diffusion:

$$\begin{aligned} &p_\theta\big(S_{0:T} \mid (I^{\text{ctx}}, \Pi^{\text{ctx}}, \Pi^{\text{tgt}})\big) \\ &= \textstyle\prod_{t=1}^T p_\theta\big(S_{t-1} \mid I_t^{\text{tgt}},\, (I^{\text{ctx}}, \Pi^{\text{ctx}}, \Pi^{\text{tgt}})\big). \end{aligned} \quad (12)$$

where the chain is driven by the reverse diffusion over target images with terminal prior $I_T^{\text{tgt}} \sim \mathcal{N}(0, \mathbf{I})$, and each S_t is obtained from I_t^{tgt} via (9).

3.3. SplatDiffuser Architecture

Setup. At each reverse diffusion step we observe a *noisy* target view $(I_t^{\text{tgt}}, \Pi^{\text{tgt}})$ and a *clean* context view $(I^{\text{ctx}}, \Pi^{\text{ctx}})$. The goal of SplatDiffuser is to predict two view-anchored sets of 3D Gaussian splats, $\mathcal{G}_t^{\text{tgt}}$ and $\mathcal{G}_t^{\text{ctx}}$, whose union $\mathcal{G}_t = \mathcal{G}_t^{\text{tgt}} \cup \mathcal{G}_t^{\text{ctx}}$ renders the target image and any additional supervision views (Sec. 3.4). Architecturally, this is realized as a two-stream DiT: one stream processes the noisy target, the other the clean context. The streams share the same block structure and parameters, but receive different camera and timestep conditioning. Figure 2 shows a detailed architecture of SplatDiffuser.

Tokenization and DiT backbone. Each input image is first mapped to a grid of latent tokens by a lightweight convolutional stem followed by a ViT-style patch tokenizer. For view $\star \in \{\text{tgt}, \text{ctx}\}$ we obtain a token sequence $\mathbf{X}^\star \in \mathbb{R}^{N_p \times d}$, where N_p is the number of patches and d the hidden dimension. Positional information is injected by either rotary embeddings or a fixed 2D sinusoidal encoding added to the tokens. The resulting sequences are processed by a stack of DiT blocks consisting of self-attention and MLP layers modulated via AdaLN-Zero conditioning.

Camera conditioning. To inject geometry and diffusion timestep into all layers, we condition each view on both the timestep and its camera pose. For a view $\star \in \{\text{tgt}, \text{ctx}\}$ we first form a *global* conditioning vector by concatenating a diffusion timestep embedding $\tau(t)$ with a camera embedding $\phi(\Pi^\star)$ derived from intrinsics and extrinsics:

$$\mathbf{c}^\star = \text{MLP}\big(\tau(t) \,\|\, \phi(\Pi^\star)\big).$$

From $\mathbf{c}^\star$, a small MLP in each block predicts a per-block scale γ_ℓ, shift β_ℓ, and residual gate ω_ℓ, which modulate the layer-normalized activations and gate the residual path via AdaLN-Zero. This "joint conditioning" treats camera parameters as an extra conditioning signal analogous to time, ensuring that both the diffusion step and the pose influence every layer.

In our final model we further convert $\mathbf{c}^\star$ into a *camera token* that is appended to the patch tokens and participates in self- and cross-attention. This token acts as a compact container for global geometric information that can be read from and written to by all spatial tokens, enabling geometry to be aggregated and redistributed across views. For some experiments we additionally augment the input with *per-pixel* camera features: intrinsics and extrinsics are converted into dense geometry (e.g., ray directions and depth priors) concatenated with the RGB channels before patch embedding. Section 4.3 compares joint-only conditioning, per-pixel features, and the camera-token variant.

Multi-view feature fusion. A key component of SplatDiffuser is the mechanism used to exchange information be-

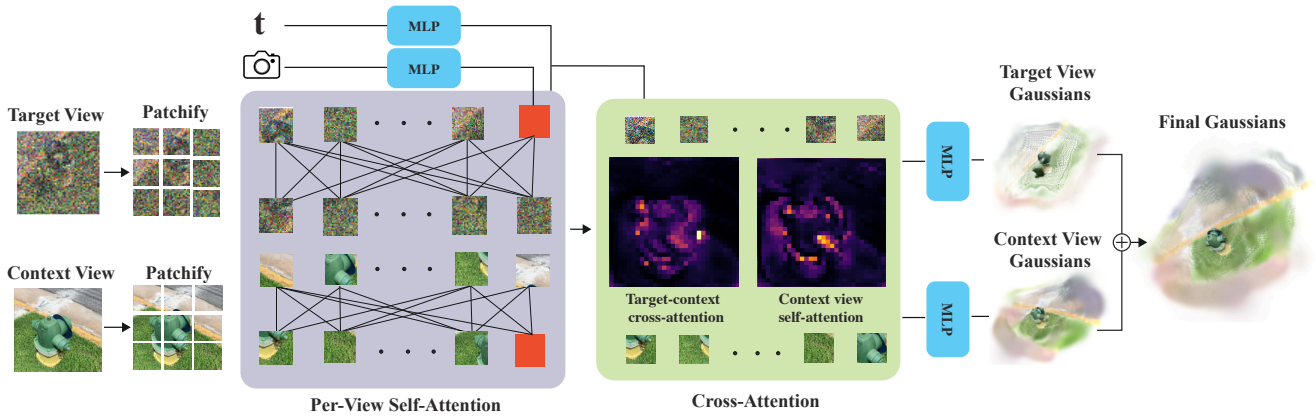


Figure 2. Model architecture of SplatDiffuser. After patchifying the target and context images, each view is encoded with per-view self-attention. A cross-attention block enables information exchange between target and context tokens, conditioned on timestep and camera features. Separate MLP heads convert the refined tokens into target-view and context-view Gaussian splats, which are subsequently fused into the final 3D Gaussian representation.

tween the noisy target and clean context streams. We consider several ways of fusing multi-view features: (i) *per-view self-attention*, where each view is processed independently without inter-view communication; (ii) *naïve view stacking*, which concatenates all tokens and applies standard self-attention over the joint sequence; and (iii) *reference attention* in the spirit of Zero123++ [37], where target tokens attend to key–value pairs computed from the context view while both streams keep their own positional encodings. These alternatives serve as baselines in our ablations (Sec. 4.3).

Our final architecture adopts an *alternating self/cross-attention* scheme. Each view first passes through several DiT blocks with self-attention and MLP (with AdaLN-Zero), building strong intra-view features. We then apply a stack of blocks that alternate self-attention and cross-attention. In a cross-attention block, queries from the target view attend to keys/values from the context view (and optionally vice versa):

$$\mathbf{X}^{\text{tgt}} \leftarrow \mathbf{X}^{\text{tgt}} + \boldsymbol{\gamma}_{\text{cross}} \odot \text{Attn}\big(\mathbf{X}^{\text{tgt}}, \mathbf{X}^{\text{ctx}}, \mathbf{M}\big),$$

where $\mathbf{M}$ is a bias encoding a soft geometry prior derived from $(\Pi^{\text{tgt}}, \Pi^{\text{ctx}})$ that steers attention toward correspondence-consistent neighborhoods, and $\boldsymbol{\gamma}_{\text{cross}}$ is a learned gate from the conditioning vector. Concretely, $\mathbf{M} \in \mathbb{R}^{N_p \times N_p}$ encodes epipolar consistency,

$$\mathbf{M}_{ij} = -\frac{d_{\text{epi}}(i,j)}{\tau},$$

where $d_{\text{epi}}(i,j)$ is the distance from the pixel location of token j to the epipolar line induced by token i under $(\Pi^{\text{tgt}}, \Pi^{\text{ctx}})$, and τ is a temperature. This softly encourages attention along epipolar lines without hard masking. Stacking a few such layers yields progressive cross-view disambiguation: tokens with reliable correspondences open their gates and aggregate context evidence, while unreliable tokens keep small gates and fall back to intra-view cues. In practice we find that a small number of such alternating layers is sufficient to capture cross-view consistency while keeping computation manageable (see Sec. 4.3 for an ablation study).

Decoding the Gaussians. For each token j at pixel $\tilde{\mathbf{u}}_j^\star = [u_j, v_j, 1]^\top$ in view $\star$, we predict *pixel-aligned* Gaussians $(d, \boldsymbol{\delta}, \mathbf{s}, \mathbf{r}, \alpha, c)$ from $\mathbf{X}_j^\star$ using a two-stage head, where d is the depth along the camera ray. $\mathbf{s}$ and $\mathbf{r}$ are the scale and rotation of a Gaussian. We first upsample the token features to a dense per-pixel feature map at the input resolution. Then a set of lightweight, disentangled projection heads operate per-pixel: a center head h_{xyz} (implemented as depth+offset) outputs $(d, \boldsymbol{\delta})$, a scale head h_{scale} outputs $\mathbf{s}$, a rotation head h_{rot} outputs $\mathbf{r}$, an opacity head h_α outputs α, and a color head h_{rgb} outputs $\mathbf{c}$. At each denoising step t, given (x_t, I^{ctx}, t), the disentangled heads produce $\mathcal{G}_t^{\text{noisy}}$ and $\mathcal{G}_t^{\text{ctx}}$; we form the union $\mathcal{G}_t$ and render the target view $\hat{I}_t = \mathcal{R}(\mathcal{G}_t, \Pi^{\text{tgt}})$. We then treat $\hat{I}_t$ as the current denoised estimate $\hat{x}_0$ and update via a deterministic DDIM step. At $t{=}0$ we output the final per-view sets and their union for rendering at arbitrary cameras.

3.4. Learning formulation

Denoising setup. We train as conditional denoising: the target RGB is diffused to x_t while the context stays clean. Let $x_t = I_t^{\text{tgt}}$, draw $t \sim \mathcal{U}\{0, \ldots, T{-}1\}$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$,

$$x_t \;=\; \sqrt{\bar{\alpha}_t}\, x_0 \;+\; \sqrt{1-\bar{\alpha}_t}\, \epsilon.$$

The network consumes $(x_t, I^{\text{ctx}}, t, \Pi^{\text{tgt}}, \Pi^{\text{ctx}})$, predicts per-view splats $\mathcal{G}^{\text{tgt}}, \mathcal{G}^{\text{ctx}}$, and we always render their union $\mathcal{G} =$

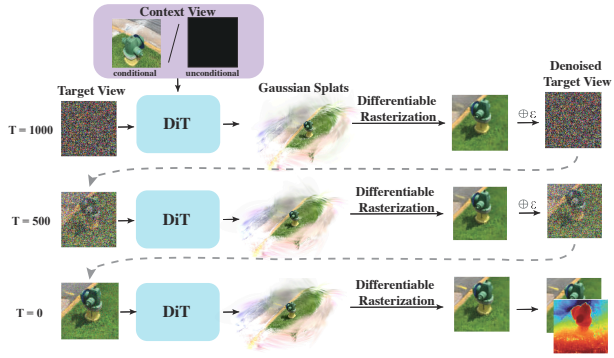


Figure 3. Overview of the denoising process in SplatDiffuser. At each diffusion timestep, the model receives a noisy target view and an optional context view and predicts a set of 3D Gaussian splats through the DiT backbone, and renders them via differentiable rasterization to obtain a denoised estimate of the target view. This estimate is then used to compute the posterior mean and sample the next noise level.

$\mathcal{G}^{\text{tgt}} \cup \mathcal{G}^{\text{ctx}}$ at supervising camera(s) $\Pi \in \mathcal{C}$:

$$\hat{I}_{\Pi} \;=\; \mathcal{R}\big(\, \mathcal{G}_\theta(x_t, I^{\text{ctx}}, t, \Pi^{\text{tgt}}, \Pi^{\text{ctx}}),\; \Pi \,\big).$$

An overview of the inference pipeline is shown in Fig. 3.

Reconstruction objective. We supervise rendered images with a weighted blend of a pixel loss and LPIPS. Let $\ell_{\text{pix}} \in \{\ell_1, \ell_2\}$. The per-view image loss is

$$\ell_{\text{img}}(\hat{I}, I) = \lambda_{\text{pix}}\, \ell_{\text{pix}}(\hat{I}, I) + \lambda_{\text{lp}}\, \text{LPIPS}(\hat{I}, I).$$

Aggregating over a set of supervised camera poses $\mathcal{C}$ per training sample,

$$\mathcal{L}_{\text{img}} = \mathbb{E}\left[\frac{1}{|\mathcal{C}|} \sum_{\Pi \in \mathcal{C}} \ell_{\text{img}}(\hat{I}_{\Pi}, I_{\Pi})\right].$$

Stage 1: $\lambda_{\text{lp}} = 0$ (pixel-only). *Stage 2:* $\lambda_{\text{lp}} > 0$ with $\lambda_{\text{pix}} = 1 - \lambda_{\text{lp}}$ (pixel+LPIPS).

Scale regularization. Following [40], we add a simple scale prior that penalizes pathological values: very large splats receive a proportional penalty (via their mean scale), while extremely small splats receive a logarithmic barrier penalty (via $-\log$ of their scale). The total regularizer is the sum of these two terms.

Total loss and schedule. Our training objective is

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{reg}}.$$

We train for 500k iterations on a single NVIDIA A100 (40 GB) with batch size 8. For each sample, we choose one context view and one target view within a pose-distance threshold, and supervise the target pose plus four additional novel poses. Details of the denoising update are described in Section 3.1. Here we simply render predictions at the selected poses and compute the losses above.

4. Experiments

We evaluate SplatDiffuser on three settings: (i) **single-view generative reconstruction** (ii) **category-level unconditional generation** (iii) **generative reconstruction with classifier-free guidance**. We compare SplatDiffuser against feed-forward 3D Gaussian Splatting approaches and diffusion-based baselines, and conduct ablation studies on cross-view fusion strategies and camera conditioning.

4.1. Experimental settings

Datasets We evaluate SplatDiffuser on both object and scene-level datasets. **CO3D** [31] is a category-level collection of in-the-wild video sequences with per-frame camera poses estimated via SfM and foreground masks. We use the official train and test splits for *Hydrants* and *Teddybears*. During testing, we follow SplatterImage's [40] fixed view sampling for consistency. We evaluate both the full image and foreground only reconstruction. **Re10K** [48] contains indoor handheld video trajectories of real estate walkthroughs with camera poses from SfM. The scenes are diverse in geometry, depth range, and appearances. We follow pixelSplat's bounded view sampling strategy [5]. The sampler selects one context view and one target view with controllable spacing. Images for both datasets are processed at 128×128 resolution.

Baselines. We compare SplatDiffuser against two representative approaches: Splatter Image [40], a feed-forward object-centric reconstruction model, and Diffusion with Forward Models (DFM) [42], a generative reconstruction framework that integrates a sparse-view feed-forward Neural Radiance Field [45] within a diffusion model. For fair comparison, we also retrain Splatter Image on full images rather than masked foregrounds, as the original implementation was limited to masked inputs. We also report results using Splatter Image trained on foreground-only inputs to assess the effect of input masking.

Metrics. We report both reconstruction and generative metrics. For reconstruction, we calculate PSNR and LPIPS scores. Since our model is a generative framework, we also compute FID and KID scores, which quantify the realism and diversity of generated samples compared to real data distributions.

4.2. Results

Reconstruction performance. We evaluate single-view conditional reconstruction on CO3D (Hydrants, Teddybears) and RealEstate10K (Re10K). Quantitative results are summarized in Tables 1–3. SplatDiffuser consistently outperforms Splatter Image retrained on full images and achieves PSNR comparable to DFM, despite DFM using two context views; DFM retains an advantage on LPIPS and

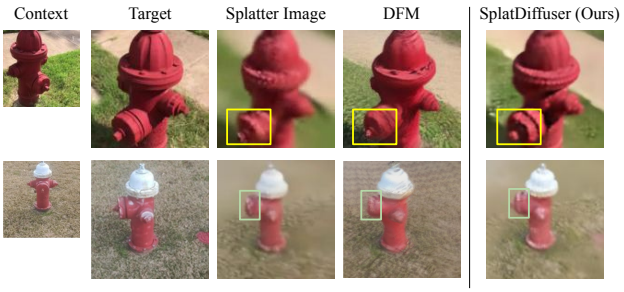


Figure 4. Single-view reconstruction on CO3D Hydrants. Given a single context image, SplatDiffuser produces sharper details and more stable geometry compared to Splatter Image and DFM. Highlighted regions indicate differences in local structure and show that our method reduces blurring and better preserves fine details.

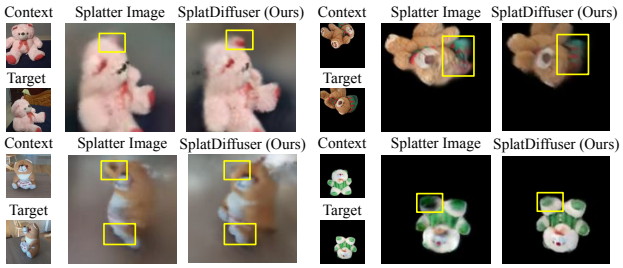


Figure 5. Single-view reconstruction on CO3D Teddybears. Since DFM is not trained on this category, we compare SplatDiffuser against Splatter Image using both full-image inputs and foreground-masked inputs. Across both settings, SplatDiffuser produces sharper details and more stable geometry, as highlighted by the marked regions.

FID. Notably, DFM requires running a full denoising process for *each* target viewpoint, resulting in inference times of approximately **10 minutes per Hydrants scene**. In contrast, SplatDiffuser requires only a single denoising loop for the entire scene—rather than one per novel view—and leverages efficient Gaussian rasterization, completing inference in **under 5 seconds** per scene. Figures 4 and 5 show qualitative comparisons. On Hydrants, SplatDiffuser produces sharper edges and more stable geometry than Splatter Image and avoids the texture smearing commonly observed in DFM. On Teddybears, where DFM is not trained, we compare against Splatter Image on both full-image and FG-masked inputs. In all settings, SplatDiffuser reconstructs finer details and maintains better cross-view consistency. Together, these results demonstrate that SplatDiffuser offers competitive accuracy at a fraction of the computational cost.

Generation performance. We further assess the generative capability of SplatDiffuser on Re10K through two complementary evaluations. First, we draw multiple samples at

Method	PSNR ↑	LPIPS ↓	FID ↓	KID ↓
Full Image				
pixelNeRF	**17.93**	0.54	180.2	0.14
Splatter Image	16.92	0.56	–	–
DFM (2 views)	17.47	**0.42**	**84.6**	**0.05**
SplatDiffuser (ours)	17.01	0.51	147.6	0.12
Foreground Only				
Splatter Image (FG)	20.49	0.16	–	–
SplatDiffuser (FG)	**22.36**	**0.15**	–	–

Table 1. Single-view reconstruction on CO3D Hydrants.

Method	PSNR ↑	LPIPS ↓	FID ↓	KID ↓
Full Image				
Splatter Image	16.39	0.58	–	–
SplatDiffuser (ours)	**16.61**	**0.54**	**132.87**	**0.08**
Foreground Only				
Splatter Image (FG)	18.07	0.27	–	–
SplatDiffuser (FG)	**19.64**	**0.25**	–	–

Table 2. Single-view reconstruction on CO3D Teddybears.

Method	PSNR ↑	LPIPS ↓	FID ↓	KID ↓
pixelNeRF	–	–	195.4	0.14
DFM (2 views)	–	–	42.84	**0.01**
DiffusionGS	**21.63**	**0.27**	**15.87**	–
SplatDiffuser (ours)	21.57	0.28	29.24	**0.01**

Table 3. Single-view reconstruction on RealEstate10K (Re10K). PSNR ↑ / LPIPS ↓ / FID ↓ / KID ↓.

CFG $= 1.0$ to analyze the diversity of purely conditional predictions; these results are shown in Figure 6. Second, we vary the classifier-free guidance (CFG) scale to study how guidance strength affects reconstruction sharpness and the model's handling of occluded regions, as illustrated in Figure 7. Together, these experiments highlight both the diversity and stability of the model's generative behavior.

Unconditional sampling. During training, we randomly drop the context image with 20% probability, enabling *fully unconditional* generation at test time. Figure 8 shows unconditional Gaussian Splat samples produced by SplatDiffuser. From noise alone, the model generates coherent RGB views with depth rendered directly from the predicted splats and supports consistent novel-view synthesis. This demonstrates that SplatDiffuser learns a meaningful 3D prior over the category distribution, rather than relying solely on conditioning information.



Figure 6. Single-view reconstruction on RealEstate10K. From a single context view, SplatDiffuser generates multiple valid interpretations of the unseen target view, resolving occluded regions in diverse yet physically plausible ways.



Figure 7. Single-view reconstruction on RealEstate10K with varying classifier-free guidance (CFG). The model remains stable across guidance strengths, with only subtle differences in sharpness and in how occluded regions are completed.

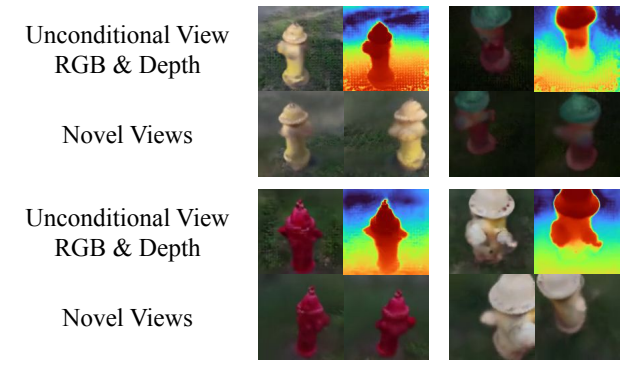


Figure 8. Unconditional 3D Gaussian Splat generation on CO3D Hydrants. SplatDiffuser produces a complete Gaussian representation from scratch; the shown RGB and depth maps are rendered from these splats. Rendering from novel viewpoints yields consistent multiview appearances.

4.3. Ablations

Multi-view feature fusion. As discussed in Sec. 3.3, SplatDiffuser can fuse target and context features in several ways. We compare (i) *per-view self-attention* (no cross-view communication), (ii) *naïve view stacking*, (iii) *reference attention* [37], and (iv) two variants of alternating self/cross-attention: target→context only, and symmetric (bidirectional) cross-attention. Table 4 reports results on CO3D Hydrants (128×128). Architectures without explicit cross-attention underperform, confirming the importance of explicit inter-view message passing.

Multi-view Fusion Strategy	PSNR ↑	LPIPS ↓
Per-view Self-Attn	16.82	<u>0.53</u>
Naïve View Stacking	16.93	0.55
Reference Attention	16.51	0.58
Target→Context Cross-Attn	<u>16.95</u>	**0.51**
Symmetric Cross-Attn	**17.01**	**0.51**

Table 4. Ablation on multi-view feature fusion strategies. Symmetric cross-attention yields the best balance between PSNR and LPIPS.

Camera conditioning. We next ablate how camera information is injected into the DiT (Sec. 3.3): (i) *joint conditioning*, (ii) *per-pixel camera features*, (iii) *camera tokenization* Results in Table 5 show that enriching the conditioning with geometric information consistently improves performance over joint conditioning alone. Per-pixel geometry already brings a noticeable gain, and the camera token further improves both PSNR and LPIPS by allowing geometry to be aggregated and broadcast across views. We therefore use the camera-token variant in all reported results.

Camera Conditioning Method	PSNR ↑	LPIPS ↓
Joint Conditioning	16.61	0.61
Per-Pixel Camera Features	<u>16.75</u>	<u>0.58</u>
Camera Token	**17.01**	**0.51**

Table 5. Ablation on camera conditioning strategies (CO3D Hydrants, 128×128). PSNR ↑ / LPIPS ↓.

5. Conclusion

We introduced SplatDiffuser, a generative framework that performs denoising diffusion directly in 3D Gaussian space, enabling diverse and view-consistent 3D reconstructions from a single image. By removing the latent bottleneck of prior approaches, our model supports end-to-end training from image supervision and, by design, avoids the detail loss that latent compression can introduce. Experiments on

CO3D and RealEstate10K demonstrate competitive reconstruction quality and controllable sampling diversity compared to deterministic and diffusion-based baselines.

Limitations. While efficient and scalable, like DFM [42], SplatDiffuser still depends on accurate camera calibration. Our experiments operate at a relatively low spatial resolution (128×128) due to GPU memory constraints; standard techniques for scaling DiT-based models, such as windowed attention or hierarchical architectures, exist but remain unvalidated for our setting. On scene-level data, renderings are blurrier than object-level results, which we attribute in part to the token-upsampling layer that maps patch tokens to per-pixel Gaussian parameters. Finally, by relying on a frustum-based Gaussian representation [5, 40], SplatDiffuser focuses primarily on scene content within the frustum. Determining the optimal Gaussian representation that supports large-scale scenarios and best aligns with denoising objectives remains an open question for future research.

References

[1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 1

[2] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2024. 1, 2

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2017. 1

[4] Yuanhao Cai, He Zhang, Kai Zhang, Yixun Liang, Mengwei Ren, Fujun Luan, Qing Liu, Soo Ye Kim, Jianming Zhang, Zhifei Zhang, Yuqian Zhou, Yulun Zhang, Xiaokang Yang, Zhe Lin, and Alan Yuille. Baking gaussian splatting into diffusion denoiser for fast and scalable single-stage image-to-3d generation. *arXiv:2411.14384*, 2025. 3

[5] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 1, 2, 6, 9

[6] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, 2024. 1, 2

[7] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. 2

[8] Xin Fei, Wenzhao Zheng, Yueqi Duan, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Jiwen Lu. Pixelgaussian: Generalizable 3d gaussian reconstruction from arbitrary views. *arXiv preprint arXiv:2410.18979*, 2024. 2

[9] Ruiqi Gao, Aleksander Hołyński, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: create anything in 3d with multi-view diffusion models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 75468–75494, 2024. 2

[10] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 1

[11] Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting. *arXiv preprint arXiv:2403.15624*, 2024. 1

[12] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1

[13] Paul Henderson, Melonie de Almeida, Daniela Ivanova, and Titas Anciukevičius. Sampling 3d gaussian scenes in seconds with latent diffusion models. *arXiv preprint arXiv:2406.13099*, 2024. 2, 3

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[15] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022. 2

[16] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024. 2

[17] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2

[18] Sergey I Kabanikhin. Inverse and ill-posed problems: theory and applications. In *Inverse and Ill-posed Problems*. de Gruyter, 2011. 1

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2

[20] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014. 2

[21] Ziwei Liao, Binbin Xu, and Steven L. Waslander. Toward general object-level mapping from sparse views with 3d diffusion priors. In *CoRL*, 2024. 2, 3

[22] Ziwei Liao, Mohamed Sayed, Steven L. Waslander, Sara Vicente, Daniyar Turmukhambetov, and Michael Firman. Complete gaussian splats from a single image with denoising diffusion models. *arXiv:2508.21542*, 2025. 2

[23] Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong Mu. Diffsplat: Repurposing image diffusion models for scalable gaussian splat generation. In *ICLR*, 2025. 3

[24] Xuyi Meng, Chen Wang, Jiahui Lei, Kostas Daniilidis, Jiatao Gu, and Lingjie Liu. Zero-1-to-g: Taming pretrained 2d diffusion model for direct 3d generation. *arXiv preprint arXiv:2501.05427*, 2025. 2, 3

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

[26] Yuxuan Mu, Xinxin Zuo, Chuan Guo, Yilin Wang, Juwei Lu, Xiaofeng Wu, Songcen Xu, Peng Dai, Youliang Yan, and Li Cheng. Gsd: View-guided gaussian splatting diffusion for 3d reconstruction. In *ECCV*, 2024. 3

[27] Muzammal Naseer, Salman Khan, and Fatih Porikli. Indoor scene understanding in 2.5/3d for autonomous agents: A survey. *IEEE Access*, 7:1859–1887, 2018. 1

[28] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 1

[29] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 2

[30] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[31] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10901–10911, 2021. 2, 3, 6

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2

[33] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems*, 2020. 1

[34] Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025. 1

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[36] Katja Schwarz, Norman Müller, and Peter Kontschieder. Generative gaussian splatting: Generating 3d scenes with video diffusion priors. In *ICCV*, 2025. 3

[37] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model, 2023. 5, 8

[38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 2

[39] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *arXiv preprint arXiv:2406.04343*, 2024. 1, 2

[40] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024. 1, 2, 6, 9

[41] Stanislaw Szymanowicz, Jason Y. Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T. Barron, and Philipp Henzler. Bolt3d: Generating 3d scenes in seconds. *arXiv:2503.14445*, 2025. 2, 3

[42] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B. Tenenbaum, Frédo Durand, William T. Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. In *NeurIPS*, 2023. 3, 4, 6, 9

[43] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *ECCV*, 2024. 2

[44] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *ECCV*, 2024. 2

[45] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 6

[46] Chubin Zhang, Hongliang Song, Yi Wei, Yu Chen, Jiwen Lu, and Yansong Tang. Geolrm: Geometry-aware large reconstruction model for high-quality 3d gaussian generation. In *NeurIPS*, 2024. 2

[47] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *ECCV*, 2024. 2

[48] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):65:1–65:12, 2018. 2, 3, 6

SplatDiffuser: Generative 3D Reconstruction from a Single Image

Supplementary Material

6. Methods Details

6.1. Architecture Details

Figure 2 in the main paper shows the architecture of our Gaussian prediction network. Each input view is an RGB image (optionally concatenated with a 3-channel ray-direction map) that is first encoded by a 7×7 stride-2 convolutional stem with 64 channels, followed by a patch-embedding layer with patch size 2 that maps features to a hidden size of 1024. The backbone is a DiT-L/2 transformer with 20 blocks, each using multi-head attention with 16 heads and an MLP expansion ratio of 4.0, combined with adaLN-Zero conditioning on the diffusion timestep. Camera poses are embedded via a harmonic embedding of the 3×4 extrinsic matrix followed by an MLP into a conditioning vector. As in Sec. 3.3, we further convert this conditioning vector into a camera token appended to the patch tokens, which participates in self- and cross-attention. The DiT output tokens are unpatched back to a feature map and passed through a set of parallel 1×1 convolutional heads, which produce the per-pixel 3D Gaussian parameters (depth/offsets, opacity, anisotropic scales, rotations, and SH color coefficients) used by the differentiable rasterizer.

6.2. Multi-view fusion attention baselines

Per-view self-attention. In this baseline, the target and context views are processed by two independent DiT streams. Each stream only uses self-attention within its own tokens, and the only interaction between views happens implicitly at the very end, when the two predicted Gaussian sets are fused for rendering. Because there is no explicit message passing, ambiguous regions in the noisy target view cannot borrow geometric cues from the clean context view. As a result, this variant behaves almost like two separate single-view models and trails both cross-attention variants in PSNR and LPIPS in Table 4.

Naïve view stacking. The naïve stacking baseline simply concatenates target and context tokens into a single long sequence and applies standard self-attention. This gives every token access to information from both views, but the model is never told which tokens are "noisy target" versus "clean context," except for the timestep modulation from adaLN-Zero. This variant brings only a marginal PSNR gain and even slightly worsens LPIPS compared to per-view self-attention, indicating that the extra flexibility is not used in a structured way.

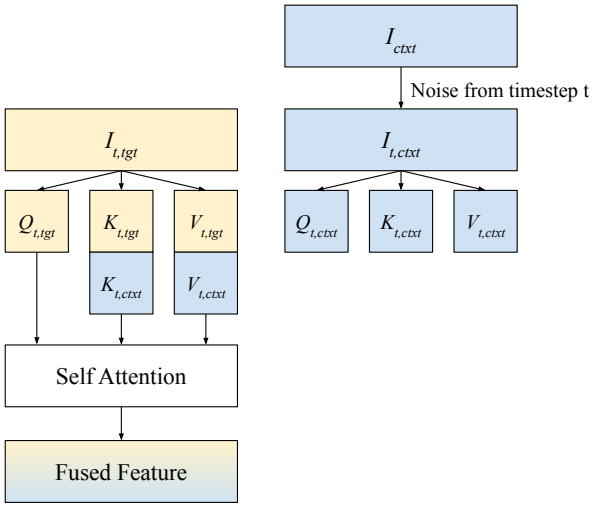


Figure 9. Reference attention baseline. The diffusion runs on the noisy target features, while a separately noised context image supplies additional keys and values; each self-attention block lets target queries attend to both target and context tokens to produce fused features.

Reference attention. Our reference attention baseline mirrors the conditioning scheme of Zero123++ (see Figure 9): the target branch becomes the main diffusion stream, while the context branch provides extra keys and values that are appended to the self-attention layers. Concretely, queries still come only from the noisy target latents, but the key/value matrices are extended with tokens from the clean context image. This allows the target to look at the context, but there is no dedicated context pathway, no geometric bias, and no way for the model to update the context features themselves based on what the target needs. In addition, this design only supports outputting Gaussian splats from the target view frustum, which limits the expressivity of the entire model.

7. Training Details

7.1. Training Schedule

We train the DiT-based Gaussian predictor for 500k iterations using AdamW with a batch size of 8, weight decay of 0.01, and betas $(0.9, 0.999)$. The learning rate follows a cosine annealing schedule with a base learning rate of 5×10^{-5}, decayed to a minimum of 1×10^{-6}. At each iteration we sample a multi-view training example, corrupt the RGB channels of the first (target) view using a DDPM-style noise scheduler at a random diffusion timestep, and feed

the resulting noisy target together with the remaining clean context views into the DiT backbone and Gaussian prediction head. To enable classifier-free guidance at inference, we apply conditional dropout during training: with probability 0.20 we zero the RGB channels of the conditioning view so the network learns both conditional and unconditional denoising. We additionally maintain an exponential moving average (EMA) of the model parameters with decay 0.9999, and use the EMA weights for evaluation and checkpointing.

7.2. View Sampling and Supervision

CO3D Dataset. For CO3D (Hydrants, Teddybears), we use the preprocessed splits from Splatter Image. We enforce a minimum temporal gap of 10 frames between the target view and the context view to encourage a sufficient multi-view baseline, then draw the $K = 4$ novel views from all remaining frames in the sequence. In validation and test, we fix the conditioning index following Splatter Image, and treat all remaining frames as targets.

Re10K Dataset We follow pixelSplat and gradually increase the separation between context and target view during training: over the first 150,000 steps, we linearly schedule the frame distance from 25 to 45.

7.3. Model Inference

At test time, we generate target views using deterministic DDIM sampling in image space, conditioned on the DiT-based Gaussian predictor. We start from isotropic Gaussian noise $x_T \sim \mathcal{N}(0, I)$ at the target view and keep the conditioning view fixed and clean. At each reverse diffusion step t, the DiT backbone takes as input the noisy target image x_t, the conditioning view, the timestep embedding, and the camera intrinsics/extrinsics, and predicts a 3D Gaussian scene representation $\hat{\Theta}_t = f_\theta(x_t, \mathcal{C}, t)$. The Gaussian splats are rendered from the target camera using our forward rendering model

$$\hat{x}_0 = \mathcal{R}\big(\hat{\Theta}_t; \Pi_{\mathrm{tgt}}\big),$$

where Π_{tgt} denotes the target camera parameters and $\mathcal{R}$ is the differentiable Gaussian splatting renderer used throughout the paper. We then convert $\hat{x}_0$ into an equivalent noise prediction

$$\hat{\epsilon}_t = \frac{x_t - \sqrt{\bar{\alpha}_t}\,\hat{x}_0}{\sqrt{1-\bar{\alpha}_t}},$$

and apply the DDIM update with a fixed noise schedule:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 \;+\; \sqrt{1-\bar{\alpha}_{t-1}}\,\hat{\epsilon}_t.$$

When using classifier-free guidance, we additionally evaluate an unconditional network $f_\theta(x_t, \emptyset, t)$ and form a guided noise prediction

$$\hat{\epsilon}_t^{\mathrm{cfg}} = \hat{\epsilon}_t^{\mathrm{uncond}} + w\big(\hat{\epsilon}_t^{\mathrm{cond}} - \hat{\epsilon}_t^{\mathrm{uncond}}\big),$$

with guidance weight w, which replaces $\hat{\epsilon}_t$ in the DDIM update.

8. Comparison to Related Works

Diffusion with Forward Models (DFM) DFM learns a conditional distribution over unobserved signals by integrating a differentiable forward model into the denoising loop, so that during training it only ever sees observations (e.g., images) but can still implicitly model the underlying 3D scene distribution. In its inverse-graphics instantiation, the signal is a NeRF-style radiance field and the forward model is differentiable volume rendering; the model is trained on tuples of posed images without ground-truth 3D scenes. At each denoising step, DFM takes one or a few clean context views and a noisy target view, predicts an updated 3D NeRF and then re-renders the target camera to form the clean-image estimate used in the diffusion update. This yields diverse, NeRF samples on CO3D and RealEstate10K using only image supervision, but inference is expensive: the authors report sampling times of up to 100 minutes for a 360° Co3D trajectory due to autoregressive sampling and NeRF volume rendering. SplatDiffuser adopts the same conceptual "diffusion + forward model" recipe, but replaces the NeRF signal with an explicit 3D Gaussian Splat representation and a feed-forward splat predictor wrapped as a projective denoiser: given a noisy target and clean context, we predict a full Gaussian scene in one denoising loop, re-render the target view, and use that render as the clean estimate in the diffusion update. This produces an explicit 3DGS scene in a single conditional diffusion chain, enabling sub–5 second sampling and real-time rasterization for arbitrary views while retaining the ability to generate diverse reconstructions from one image.

CompleteSplat CompleteSplat also tackles single-image generative 3D reconstruction with Gaussian splats, but does so via latent diffusion over Splatter-Image representations. The goal is to model a distribution over complete 3D scenes, including occluded and out-of-frustum regions, conditioned on one RGB image. Because there is no large-scale dataset of scene-level Splatter Images, the authors introduce a Variational AutoReconstructor that learns a latent space of Splatter Images from posed images alone, using differentiable rendering as a forward model; diffusion is then trained in this latent space and decoded back to full Gaussian scenes. Like SplatDiffuser, training relies only on posed images and a differentiable rasterizer, and the method yields diverse, 360°-complete Gaussian scenes from a single view in one denoising loop. The key contrast is that CompleteSplat models a distribution over latent Splatter codes and therefore requires a separate VAE-style reconstructor and two-stage optimization, while SplatDiffuser learns the 3DGS distribution directly, without a la-

tent bottleneck: our diffusion model operates entirely in the 3D Gaussian Splat domain via a projective denoiser, which avoids decoding errors and the detail loss that latent decoding can introduce, at the expense of a slightly heavier per-sample denoiser.

DiffusionGS DiffusionGS "bakes" 3D Gaussian Splatting directly into the denoiser to obtain a single-stage image-to-3D diffusion model for object generation and scene reconstruction from a single view. The method takes a clean condition view and multiple noisy views, and at every diffusion step predicts multi-view pixel-aligned Gaussian primitives, which are merged into 3D Gaussian point clouds and supervised only through rendered multi-view images using a differentiable rasterizer. DiffusionGS combines this with a scene–object mixed training strategy and a new camera-encoding scheme (RPPC) to scale to both object-centric assets and large RealEstate10K scenes, achieving strong PSNR/FID results while running in 6 seconds on a single A100. Compared to SplatDiffuser, DiffusionGS is a fully 3D diffusion model over multi-view images whose denoiser always emits 3D Gaussians at each step, whereas SplatDiffuser applies conditional diffusion over a single noisy target image and uses a feed-forward Gaussian predictor as a projective denoiser. Our design keeps the denoising state in image space but the generative signal in explicit 3DGS, yielding a simpler two-view DiT architecture and allowing us to retain feed-forward-like efficiency while still learning a distribution over 3D Gaussian scenes from images alone.

Bolt3D Bolt3D also outputs explicit 3DGS scenes and runs in seconds, but it follows a hybrid pipeline that couples multi-view latent diffusion with a regression head to Gaussian splats. Given one or more posed input images, Bolt3D first learns a latent diffusion model over multi-view RGB images and dense pointmaps obtained by running a separate geometric reconstructor (MASt3R) on large multi-view datasets; the diffusion model samples multi-view image+pointmap latents, which are decoded to images and depth-like pointmaps and then regressed to Splatter-Image representations that are fused into a 3DGS scene. This design leverages strong 3D supervision (MASt3R pointmaps) and multi-view coverage, and can generate text- or image-conditioned 3D scenes in under 7 seconds on a single GPU. In contrast, SplatDiffuser does not rely on any precomputed 3D geometry or external teacher; like DFM and CompleteSplat it is trained purely from 2D images and a differentiable renderer, but—unlike Bolt3D—models the 3DGS distribution directly rather than via a latent multi-view diffusion plus regression. Table 6 gives an overall comparison of the methods in terms of key design choices and inference time.

Method	Single-view conditioning	Denoises explicit 3DGS	Latent 3D space	Uses external 3D geometry	Typical inference time per scene
SplatDiffuser (ours)	Yes	Yes	No	No	≈ 5 s
DFM	Yes (1-2 views)	No (NeRF)	No	No	≈ 10 min
DiffusionGS	Yes	Yes	No	No	≈ 6 s
CompleteSplat	Yes	No	Yes	No	< 3 s
Bolt3D	Yes	No	Yes	Yes	< 7 s

Table 6. Comparison of methods by key design choices and per-scene inference time.