Splatting the Invisible: Geometry and Appearance Scene Completion from Sparse Views

Wesley Khademi
Oregon State University
khademiw@oregonstate.edu

Li Fuxin
Oregon State University
fuxin.li@oregonstate.edu

Abstract

Novel view synthesis methods such as 3D Gaussian Splatting degrade under sparse-view settings, suffering from an inability to extrapolate beyond observed regions. To enable high fidelity reconstruction and rendering from sparse views, inpainting of sparsely observed areas and completion of occluded regions is required. Unlike existing approaches that perform these tasks using 2D inpainting diffusion models or geometric-only 3D completion models, we propose a fully 3D scene completion framework, called InvisiSplat, that performs geometry and appearance completion directly in 3D. Our method consists of a Gaussian Splat VAE which learns a sparse point cloud latent space that predicts Gaussian Splat parameters from the latents, as well as a two-step flow model that learns to generate samples conditioned on the point cloud latents, different from other approaches that use tokens without 3D coordinates. Our flow model utilizes alternating local 3D attention and global attention, balancing the need for precise local geometry and large-scale spatial context. Across several indoor datasets, we demonstrate that our method produces $33\% - 93\%$ higher fidelity 3D geometry over prior work while attaining competitive performance on appearance metrics when compared to existing approaches.

1 Introduction

Every day, humans effortlessly form rich spatial imaginations of the world around them. We manipulate objects that are only partially visible, infer the full shape of objects that are occluded, and navigate spaces that are only partially revealed from our current viewpoint. This remarkable ability to complete geometry and appearance from sparse observations enables robust interaction with complex, cluttered environments. In contrast, current 3D reconstruction systems are largely limited to faithfully reconstructing visible parts of the scene [1, 2, 3, 4, 5, 6, 7]. Enabling machines to infer the invisible—to reconstruct both the underlying 3D geometry and its appearance from sparse views with large occlusions and limited sensing coverage—remains a fundamental challenge. There is some work that approaches this task as 2D video generation [8, 9, 10], but it struggles with multi-view consistency, especially in the unseen parts.

Existing 3D completion methods, however, remain limited for this task. Semantic scene completion [11, 12, 13, 14] predominantly focuses on geometry alone, predicting a complete occupancy grid of the scene. As a result, the completed scenes cannot directly support view synthesis or rendering tasks. On the other hand, instance-level point completion methods are typically designed for a single, isolated object with canonical alignment or category priors [15, 16, 17, 18, 19]. While the recent work [20] extends instance-level point completion to multi-object scenes, it completes each object separately after instance segmentation, which limits the types of instances it can complete and makes it difficult to complete articulated instances that have parts that can be attached/detached.

In this work, we propose a unified geometry and appearance scene completion framework that operates directly in 3D space. Instead of decomposing the scene into predefined object instances or predicting

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

geometry alone, our method models the entire environment as a coherent 3D representation and jointly infers structure and appearance from sparse RGB-D observations. By reasoning natively in 3D, our framework naturally handles occlusions, viewpoint variation, and complex spatial arrangements, enabling plausible completion of large unseen regions as well as photorealistic novel view synthesis, while preserving consistency with observed regions.

A key novelty in InvisiSplat is to utilize tokens on point clouds with explicit 3D coordinates to condition the generative model. This is important for preserving the partial input geometry, as well as completing large missing regions in a structurally coherent manner. Thus, we first encode the observed point cloud into a point cloud latent representation, which is used to condition a point-based flow transformer. Preserving the 3D point coordinates allowed us to alternate 3D local attention and global attention to capture both detailed geometric interactions as well as long-range spatial dependencies. This combination is crucial for completing large missing regions in a structurally coherent manner. Additionally, we employ loss weighting to prioritize semantically salient regions, encouraging accurate object reconstruction while maintaining global scene consistency. Overall, our contributions are summarized as follows:

- We propose a unified framework for joint 3D geometry and appearance scene completion from sparse views without instance decomposition, enabling coherent reasoning over complex multi-object environments.
- We propose a geometry-aware point cloud latent for conditioning our generative completion model, allowing for precise preservation of partial input geometry in completions.
- We introduce a point-based flow transformer featuring alternating local and global attention to capture both fine-grained geometry and long-range spatial dependencies in completions.
- We design a weighted flow-matching objective to prioritize semantically salient regions, improving object-level realism and multi-view consistency.

Extensive experiments on 3D-Front, ScanNet++, and Replica demonstrate **33 - 93%** improvement in geometry metrics while achieving competitive rendering performance, establishing a new state-of-the-art in 3D scene-level completion of indoor scenes.

2 Related work

Novel view synthesis Neural Radiance Fields [1] and 3D Gaussian Splatting (3DGS) [2, 3] achieve high-quality novel view synthesis from multi-view imagery, but often degrade under sparse inputs. To address this, regularization techniques [21, 22, 23, 24, 25] and the use of generative priors [26, 27, 28] have been proposed to reduce artifacts and better constrain geometry and appearance. However, these methods remain limited to interpolation and cannot extrapolate or complete unobserved regions.

Multi-view [29, 30] and video diffusion models [31, 32, 33, 34] have also been used for novel view synthesis. However, such approaches lack 3D geometry and often suffer from limited multi-view consistency. Other works [35, 36, 37] propose to use video diffusion models for generating inputs to a large reconstruction model; however, inconsistencies in the synthesized views can lead to artifacts in the reconstruction. To improve multi-view consistency, SpatialCrafter [8], MVGenMaster [10], and ViewCrafter [9] first build a reconstruction of the scene from input views and warp this reconstruction to test views as initialization to their inpainting diffusion models. This improves consistency in observed regions; however, consistency in newly generated, unobserved regions is still not guaranteed.

3D scene generation In recent years, the generation of both 3D geometry and appearance from text and images has been well explored [38, 39, 40, 41, 42, 43, 44, 45, 46], but has mostly been limited to generating single objects, which have much less complexity compared to 3D scenes where multiple objects can adopt an arbitrary configuration. [47, 48, 49] propose to learn diffusion models for scene-scale 3D scene generation; however, these works only generate 3D geometry and do not support rendering. To enable realistic scene generation, some works have explored generating both geometry and appearance. L3DG [50] and Can3Tok [51] propose to learn a diffusion model on a latent spaces over scenes' Gaussian Splat parameters to perform unconditional generation or text/image-to-3DGS generation. WorldGrow [52] adapts SLATs [46] for geometry and appearance generation, but their goal is infinite world generation rather than completion. Unlike these works, our work aims to generate scene completions from partial scans of scenes.

3D scene completion Recent 3D completion methods have focused on producing the complete 3D geometry of objects in scenes [53, 54, 55, 20]. Alternatively, some works have proposed methods for recovering both the complete 3D geometry and appearance of objects within scenes from a single image [56, 57, 58]. However, these approaches are typically limited to only completing objects in pre-defined categories.

RGBD2 [59] was the first to tackle geometry and appearance completion of scenes from sparse RGB-D views. Starting from an incomplete textured mesh, their method iteratively completes it by rendering and inpainting RGB-D images from novel views which are used for mesh fusion. GenRC [60] improved upon RGBD2 by performing panoramic RGB-D inpainting to enhance multi-view consistency. RI3D [61], GenFusion [62], and G4Splat [63] iteratively complete an initial 3DGS reconstruction by using 2D diffusion models to inpaint and repair rendered views to re-optimize the 3DGS scene with. In contrast to these works, we complete geometry and appearance directly in 3D space rather than image space, yielding stronger multi-view consistency and higher fidelity geometry.

3 Method

Given a sparse set of RGB images $\{I_i\}_{i=1}^{V_{sparse}}$ and corresponding depth maps $\{D_i\}_{i=1}^{V_{sparse}}$ with poses $\{T_i\}_{i=1}^{V_{sparse}}$, our method aims to predict the complete geometry and appearance of the scene as a set of Gaussian Splat parameters $\mathcal{G} \in \mathbb{R}^{N_g \times p}$, where N_g is the number of Gaussians and p is the number of parameters per Gaussian. Note that the difficulty of this problem lies in that both the geometry and the appearance of the occluded regions need to be generated. Prior work [60, 62, 63] usually rely on image or video diffusion models to inpaint missing regions. Because completion occurs in 2D, they usually generate nice-looking renderings but struggle to produce images that are multi-view consistent and geometrically coherent with observed geometry.

Different from these works, we propose to first generate *sparse complete geometry*, which is a sparser point cloud with the complete geometry, and then generate features on each point of this sparse complete point cloud. These features can then be decoded into a dense point cloud with per-point Gaussian Splat parameters. This design offers several advantages: (1) reasoning about occluded regions without needing to define camera trajectories, (2) inherent multi-view geometric consistency, and (3) straightforward preservation of partial input geometry. Empirically, we have observed that this approach preserves and completes geometry much better than video diffusion approaches.

Our proposed algorithm *InvisiSplat* consists of the following four steps (Fig. 1):

1. Encode observed information: We utilize a point cloud encoder to obtain geometric features, and backprojected DINOv2 features from each view to obtain appearance features of the visible partial point cloud.

2. Sparse complete geometry generation: a point flow model based on our local-global flow transformer generates the complete 3D geometry of the scene as a sparse point cloud.

3. Latent feature generation: a structured latent flow model based on our local-global flow transformer generates per-point latent features on the sparse complete point cloud.

4. Decoding: A trained Gaussian splat decoder decodes the latent features on each point into predicted Gaussian splatting parameters.

InvisiSplat is trained in two stages. In the first stage, we train for steps 1 and 4: namely, a point-based variational autoencoder (VAE) that takes a point cloud as input and, through an encoder-decoder process, aims to reconstruct the point cloud as well as predict per-point Gaussian parameters so that the scene can be rendered from any viewpoint. This VAE has a point cloud latent with 3D coordinates and a feature vector for every point, which is used to condition the flow models for scene completion in the second stage, different from prior work that condition with implicit tokens or images.

In the second stage, we train for steps 2 and 3: first a point flow model to generate the sparse complete geometry conditioned on the point cloud latent that was encoded from partial views. From the sparse complete geometry, a second structured latent flow model generates per-point latent features, which are decoded with the VAE decoder to predict splatting parameters during inference time (Fig. 1).

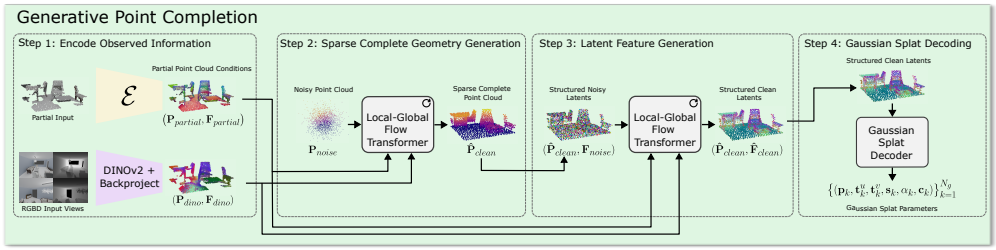

Figure 1: InvisiSplat pipeline. Input point cloud and RGB views are encoded through our VAE encoder $\mathcal{E}$ and a DINOv2 encoder, respectively, to be used as conditions for our two-step generation model. First (Step 2), our point flow model generates the sparse complete 3D geometry from the partial conditions. Then (Step 3), our latent flow model generates per-point appearance features for the predicted geometry, which can be decoded to Gaussian Splat parameters for rendering.

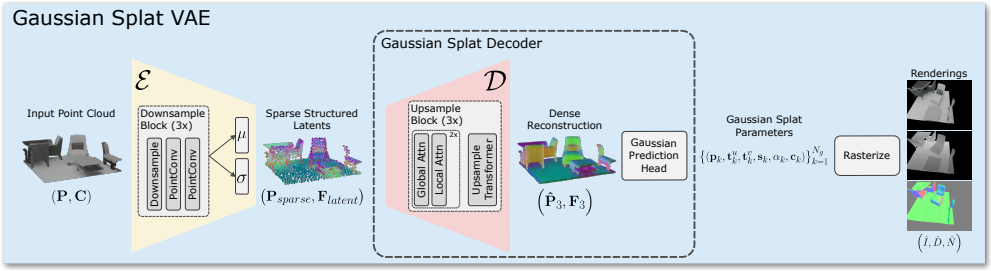

Figure 2: VAE Pretraining of Feedforward Gaussian Splatting

3.1 Gaussian Splat VAE

The goal of our Gaussian Splat VAE (Fig. 2) is to learn a structured latent space that can be decoded to Gaussian Splat parameters, from which we can train a generative model to sample. We design our VAE to be trained directly from posed RGB-D data, as opposed to optimizing per-scene 3DGS representations as done in [50, 51] because of much better scalability. While our scene completion framework takes sparse RGB-D inputs, our VAE is trained from a dense set of posed RGB-D images $\{(I_i, D_i, T_i)\}_{i=1}^{V_{dense}}$, since during inference the decoder needs to render **completed** point clouds (Step 4). These dense views are lifted to 3D to produce a point cloud with points $\mathbf{P}$ and colors $\mathbf{C}$ from which our VAE decodes a set of Gaussian Splat parameters $\mathcal{G}$ and renders them for training.

Encoder We design our encoder $\mathcal{E}$ to produce sparse structured latents from our input point cloud, $(\mathbf{P}_{sparse}, \mathbf{F}_{latent}) = \mathcal{E}(\mathbf{P}, \mathbf{C})$, where $\mathbf{P}_{sparse} \in \mathbb{R}^{N_{sparse} \times 3}$ is a coarse representation of $\mathbf{P}$ and $\mathbf{F}_{latent} \in \mathbb{R}^{N_{sparse} \times d}$ are the corresponding per-point features. Our encoder is based on [20], which consists of a series of downsampling layers and PointConv [64] layers to interpolate and aggregate point features. The mean μ and standard deviation σ are predicted by two linear layers from the final downsampled per-point features, forming a Gaussian distribution from which the features $\mathbf{F}_{latent}$ are sampled. A standard KL term for VAEs is added to our loss to encourage a compact latent space.

Decoder We introduce a decoder $\mathcal{D}$ that aims to reconstruct a dense featurized point cloud from sparse structured latents, $(\hat{\mathbf{P}}_3, \mathbf{F}_3) = \mathcal{D}(\mathbf{P}_{sparse}, \mathbf{F}_{latent})$, where $\hat{\mathbf{P}}_3 \in \mathbb{R}^{N_{dense} \times 3}$ is a dense reconstruction of our input point cloud $\mathbf{P}$ and $\mathbf{F}_3 \in \mathbb{R}^{N_{dense} \times C}$ are its corresponding per-point features. Our decoder progressively upsamples our sparse structured latents through a series of 3 upsampling blocks. In each upsampling block, features are processed by alternating global and local attention layers before both points and features are upsampled by a factor r using an Upsample Transformer from [17]. To supervise our reconstructions, we use the Chamfer distance (CD) loss:

$$\mathcal{L}_{recon} = \sum_{i=1}^{3} CD(\hat{\mathbf{P}}_i, \mathbf{P}_i) \tag{1}$$

where $\hat{\mathbf{P}}_i$ is the reconstruction at each level of the decoder and $\mathbf{P}_i$ is the input point cloud $\mathbf{P}$ resampled to the same number of points as the reconstruction $\hat{\mathbf{P}}_i$.

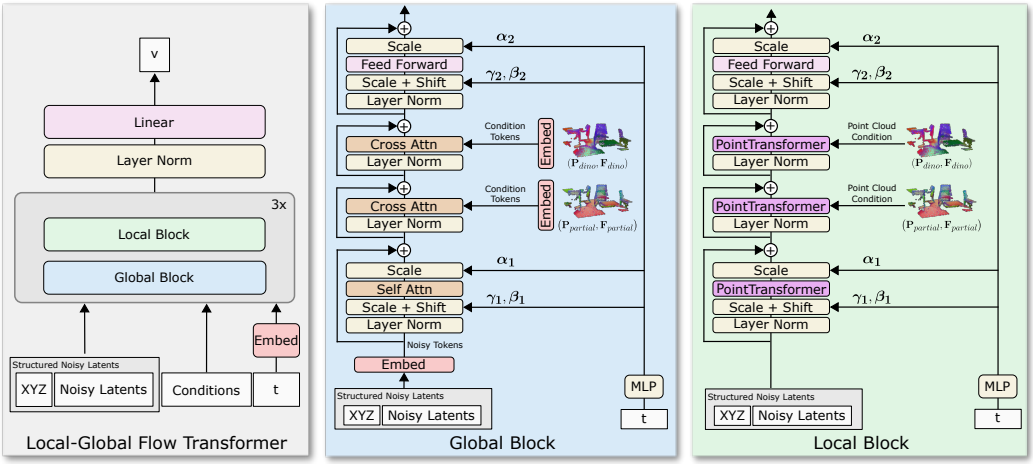

Figure 3: Overview of our Local-Global Flow Transformer, where Global Blocks and Local Blocks are alternated to capture both the global context and local 3D geometry.

Gaussian Prediction Head To recover appearance, we decode a set of Gaussians $\mathcal{G} \in \mathbb{R}^{N_g \times p}$ which can be rendered via Gaussian Splatting [2]. We choose to use 2DGS [3] as our Gaussian splat parameterization, and predict Gaussian parameters from the final upsampled point cloud $(\hat{\mathbf{P}}_3, \mathbf{F}_3)$ via MLPs. We supervise our Gaussian Prediction Head by rendering RGB images $\hat{I}$, depth maps $\hat{D}$, and normal maps $\hat{N}$ and computing rendering losses against the ground truth RGB images I, depth maps D, and normal maps N. Our final loss for VAE training is defined as:

$$\mathcal{L}_{render} = \mathcal{L}_{recon} + \mathcal{L}_{rgb}(\hat{I}, I) + \lambda_{per}\mathcal{L}_{per}(\hat{I}, I) + \lambda_d \mathcal{L}_d(\hat{D}, D) + \lambda_n \mathcal{L}_n(\hat{N}, N) + \lambda_{KL}\mathcal{L}_{KL} \quad (2)$$

where $\mathcal{L}_{recon}$ is defined in eq. (1), $\mathcal{L}_{rgb}$ is the MSE loss, $\mathcal{L}_{per}$ is the LPIPS loss, $\mathcal{L}_d$ is an L1 depth loss, $\mathcal{L}_n$ is a cosine similarity loss on the normal vectors, and $\mathcal{L}_{KL}$ is the KL regularization on our latent space. For our experiments, we set $\lambda_{per} = 10^{-1}$, $\lambda_d = 10^{-1}$, $\lambda_n = 10^{-1}$, and $\lambda_{KL} = 10^{-3}$.

3.2 Generative point completion

To produce scene completions, we introduce a two-stage flow model that learns to generate samples from our VAE's point cloud latent space when conditioned on sparse RGB-D views (Fig. 1). This includes the point flow model (step 2 in Fig. 1) which generates the sparse complete geometry from the output of the encoder, as well as the structured latent flow model (step 3 in Fig. 1) which generates the appearance features on the sparse complete geometry. Both of them share the same conditioning features from both an RGB backbone and the point cloud encoder, as well as the structure of a local-global flow transformer. Hence, we describe these first before discussing the two flow models.

Conditioning for the flow models Unlike previous 3D generation works [42, 44, 46] that condition on tokens without explicit 3D coordinates, a main novelty in our approach is to condition our flow models on point cloud latents. Specifically, we use our point cloud encoder trained in Sec. 3.1 which produces latents that have enough information for point cloud reconstruction and have explicit 3D locations for each latent. Conditioning on point cloud latents allows us to introduce 3D local attention as part of our flow transformer, which we found to be beneficial for modeling the local 3D geometry of the missing regions as well. Hence, we lift our sparse RGB-D input views to 3D and encode the partial input point cloud through the frozen VAE encoder $\mathcal{E}$ to produce a sparse featurized point cloud with points $\mathbf{P}_{partial}$ and features $\mathbf{F}_{partial}$.

To capture appearance and semantic features that come from large-scale 2D pre-training, we extract feature maps from our RGB images using DINOv2 [65] and lift them to 3D to produce a second point cloud with points $\mathbf{P}_{dino}$ and features $\mathbf{F}_{dino}$. After DINOv2 downsampling this is around the same resolution as the point cloud encoder after point cloud downsampling. The features from a 2D foundation model allow for better appearance reconstruction since the 2D backbone has seen a significantly larger amount of appearances in its training data than the 3D backbone.

Local-global flow transformer Conditional flow matching (CFM) [66] learns a time-dependent velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t)$ that follows a prescribed probability path between Gaussian noise $\mathbf{x}_0 \sim$

$\mathcal{N}(0, I)$ and data $\mathbf{x}_1 \sim p_{data}$. The probability path is typically defined via linear interpolation $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$, and the velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t)$ is learned by minimizing the CFM objective:

$$\mathcal{L}_{CFM} = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1} \|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2 \tag{3}$$

We use eq. (3) for training our flow models and note that data samples $\mathbf{x}_1$ are defined per setting. For our point flow model, $\mathbf{x}_1$ is a point $\mathbf{p} \in \mathbb{R}^3$ from a complete point cloud $\mathbf{P}_{clean}$, while for our latent flow model, $\mathbf{x}_1$ is a latent feature $\mathbf{f} \in \mathbb{R}^d$ from a VAE point cloud latent $\mathbf{F}_{clean}$. At inference time, we generate samples by solving the ODE $\frac{d}{dt}\mathbf{x}_t = \mathbf{v}_\theta(\mathbf{x}_t, t)$ from $t = 0 \to 1$ using the Euler method.

We model our velocity field $\mathbf{v}_\theta$ as a transformer. Our architecture shares similarities with DiT [67]. However, we observe that, on scene level data, global attention alone is only able to capture the coarse structure of objects but lacks smooth local geometry. To address this, we propose an alternating 3D global and local attention scheme in our flow transformer instead of the typical full global attention. Local attention allows our noisy points to refine its structure by querying local neighborhoods of the partially observed geometry, which we find helps our model recover sharper geometric details (see Fig. 7 for qualitative comparisons). We present an overview of our flow transformer in Fig. 3.

Our Local-Global Flow Transformer takes the following as input: a timestep t, encoded partial observations $\mathcal{C} = \{(\mathbf{P}_{partial}, \mathbf{F}_{partial}), (\mathbf{P}_{dino}, \mathbf{F}_{dino})\}$, and a set of structured noisy latents consisting of XYZ point coordinates and per-point noisy latents. Noisy latents are processed through a series of 3 alternating global and local attention blocks before being passed through a final linear layer to predict velocity. Within each block, self-attention is first performed, followed by cross-attention layers with the partial point cloud from the point cloud encoder, and then cross-attention with the partial point cloud from the RGB encoder. In our local blocks, we compute attention over local point neighborhoods using PointTransformer [68], which utilizes relative position encodings to learn detailed point geometry. Time embeddings are integrated through Adaptive Layer Norm as in [67].

Point flow Given the partially observed scene, our point flow model generates a sparse point cloud representing a coarse version of complete scene geometry. Our flow model operates directly in the point cloud space and we parameterize the velocity field with our local–global flow transformer. Since standard flow matching can induce large transport costs and higher sampling trajectory curvatures for point clouds [69], we follow previous works [70, 71, 72, 69] and adopt Optimal Transport (OT) flows between Gaussian noise $\mathbf{P}_{noise}$ and complete target point cloud $\mathbf{P}_{clean}$.

Our complete target point clouds $\mathbf{P}_{clean}$ used for training are obtained through furthest point sampling from dense point clouds. This typically biases sampling towards the background, with more points being on walls and floors instead of the foreground objects (e.g., chairs). To counteract this, we introduce a weighting on the CFM objective in eq. (3) that boosts foreground object reconstruction. In particular, for each point $\mathbf{p}_1^i \in \mathbf{P}_{clean}$, we assign a weight $w^i = w_{fg}^i \cdot w_{density}^i$ where:

$$w_{fg}^i = \begin{cases} \max(1, (N_{bg}/N_{fg})^\alpha) & \text{if } \mathbf{p}_1^i \text{ in foreground} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

$$w_{density}^i = \left(\frac{1}{|N(\mathbf{p}_1^i)|} \sum_{j \in N(\mathbf{p}_1^i)} \|\mathbf{p}_1^i - \mathbf{p}_1^j\|_2 \right)^{beta} \tag{5}$$

where N_{bg} and N_{fg} are the number of background and foreground points in $\mathbf{P}_{clean}$, $N(\mathbf{p}_1^i)$ corresponds to the indices of the K-nearest neighbors around point $\mathbf{p}_1^i$, and α and β are scalars. Intuitively, w_{fg}^i boosts the importance of foreground points in $\mathbf{P}_{clean}$, while $w_{density}^i$ prioritizes low density regions, which typically correspond to thin structures (e.g., legs of chairs) in sparse point clouds.

Structured latent flow Our latent flow model generates per-point appearance features for the complete geometry predicted by the point flow model via the local-global flow transformer. During training, we obtain clean latents $\mathbf{F}_{clean} \in \mathbb{R}^{N_{sparse} \times d}$ by passing a complete point cloud through our frozen VAE encoder to produce structured latents $(\mathbf{P}_{clean}, \mathbf{F}_{clean})$. Unlike the point flow model, here we use standard flow matching instead of OT Flows. Noisy latents at timestep t are created by interpolating between $\mathbf{F}_{clean}$ and Gaussian noise $\mathbf{F}_{noise} \in \mathbb{R}^{N_{sparse} \times d}$. During training, noisy latents are assigned to each point from the ground-truth sparse complete geometry $\mathbf{P}_{clean}$ as input to the local-global flow transformer. At inference, full Gaussian noise $\mathbf{F}_{noise}$ is attached to the predicted sparse complete geometry $\hat{\mathbf{P}}_{clean}$ to be used as input for denoising. After the clean appearance latents are generated, the pre-trained decoder (Sec. 3.1) can decode them into Gaussian Splat parameters and render them from any viewpoint.

4 Experiments

Implementation details For training our Gaussian Splat VAE, we use a batch size of 2 and a learning rate of $1e-4$ which is linearly decayed by a rate of 0.75 every 50 epochs. We train for 400 epochs on a single NVIDIA H100, which takes approximately 36 hours. For training our flow models, we use a batch size of 15 and a learning rate of $1e-4$. We train each flow model on 3 NVIDIA L40S, with the point flow model being trained for 2000 epochs and the latent flow model being trained for 1500 epochs, taking approximately 4 and 3 days, respectively. At inference, we use 60 sampling steps for generating geometry and appearance with our flow models. We refer readers to our appendix for more model architecture and training details.

Datasets We jointly train our model on a mix of real data from ScanNet++ [73] and synthetic data from the 3D-FRONT dataset [74, 75], where we use the official ScanNet++ train split consisting of 856 scenes and randomly select another 595 rooms from 3D-FRONT. For evaluation, we select 15 scenes from the ScanNet++ validation set and 10 unseen rooms from 3D-FRONT. We further perform evaluation on an additional 4 scenes from the Replica dataset [76] to evaluate our method's generalization capabilities. For each evaluation scene, we select 3 to 4 input views, which when lifted to 3D produce a point cloud with sufficiently missing geometry compared to the ground truth meshes. We further select 3 to 4 test views per scene, ensuring that a portion of each view's image corresponds to a region of missing geometry. In this way, we can evaluate our methods ability to do both interpolation and extrapolation.

Metrics We evaluate our approach in terms of geometry, appearance, and multi-view consistency. For geometry, we compute Chamfer distance (CD) between predicted and ground-truth point clouds. Partial reconstruction quality (P-OCD) is measured using one-sided CD from the partial input to the prediction. To isolate completion evaluation to areas missing from the partial input, we extract missing regions from the ground-truth point cloud and compute its one-sided CD to the prediction (M-OCD). We also report L1 error on rendered depth maps. For appearance, we use PSNR, SSIM, and LPIPS. For multi-view consistency, we apply MEt3R [77] to rendered views, measuring similarity in overlapping regions to assess consistency across viewpoints.

Baselines We compare our approach against several sparse view scene completion methods. GenRC [60] iteratively builds a complete textured mesh by leveraging StableDiffusion [78] for image and panorama inpainting. ViewCrafter [9] uses an inpainting video diffusion model for novel view synthesis trained on a large scale video dataset from DL3DV [79] and RealEstate10k [80]. GenFusion [62] and G4Splat [63] propose iterative reconstruction and repairing algorithms using 2DGS [3] and a video diffusion model trained on large scale video datasets [79, 81].

4.1 Results

Geometry quality We quantitatively evaluate 3D reconstruction and completion on the 3D-FRONT dataset in Table 1. GenRC performs poorly in both partial reconstruction (P-OCD) and completion (M-OCD), and further finetuning of the method on 3D-FRONT does not improve geometry quality. GenFusion achieves strong partial reconstruction but struggles to complete missing geometry, reflected by its high M-OCD. Conversely, G4Splat attains stronger completion but weaker partial reconstruction, with lower M-OCD yet high P-OCD. In contrast, our method performs strongly across all metrics, improving completion (M-OCD) **by 44%** and overall reconstruction (CD) **by 33%** over the best baseline. It is also the only method to produce accurate, complete rendered depth maps, reducing L1

Table 1: Quantitative results on the 3D-FRONT dataset. Methods below the separator use 3D-FRONT training data; † denotes fine-tuned baselines.

	Geometry				Appearance			Consistency
	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
GenRC [60]	0.1149	0.0166	0.1198	0.3735	16.53	0.7073	0.4469	0.5522
ViewCrafter [9]	*	*	*	*	17.51	0.7447	0.3509	0.7087
GenFusion [62]	0.0612	**0.0003**	0.0837	0.3283	15.82	0.7342	0.4832	0.7204
G4Splat [63]	0.0739	0.0183	0.0599	0.3186	16.10	0.7231	0.4384	0.5880
GenRC† [60]	0.1132	0.0210	0.1156	0.3778	17.22	0.7452	0.4181	0.5663
ViewCrafter† [9]	*	*	*	*	18.16	**0.7694**	**0.3429**	0.6435
InvisiSplat (Ours)	**0.0408**	**0.0003**	**0.0338**	**0.0931**	**19.05**	0.7505	0.3581	**0.5007**

Table 2: Quantitative results on the ScanNet++ dataset.

	Geometry				Appearance			Consistency
	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
GenRC [60]	0.0984	0.0114	0.1776	0.1348	**17.24**	**0.7067**	0.3820	**0.4844**
ViewCrafter [9]	*	*	*	*	14.04	0.5954	0.5111	0.7805
GenFusion [62]	0.0375	**0.0002**	0.0800	0.2089	15.63	0.6641	0.5027	0.6955
G4Splat [63]	0.0432	0.0086	0.0491	0.0941	16.79	0.6894	0.4154	0.5377
InvisiSplat (Ours)	**0.0089**	**0.0002**	**0.0090**	**0.0544**	16.82	0.6931	**0.3696**	0.4959

Table 3: Quantitative results when generalizing to the Replica dataset.

	Geometry				Appearance			Consistency
	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
GenRC [60]	0.0582	0.0163	0.0653	0.1764	21.86	0.8127	0.3662	**0.4757**
ViewCrafter [9]	*	*	*	*	20.49	0.8021	**0.3295**	0.5637
G4Splat [63]	0.0115	0.0013	0.0226	0.0522	**24.21**	0.8386	0.3693	0.5216
InvisiSplat (Ours)	**0.0030**	**0.0002**	**0.0015**	**0.0390**	24.01	**0.8442**	0.3365	0.4777

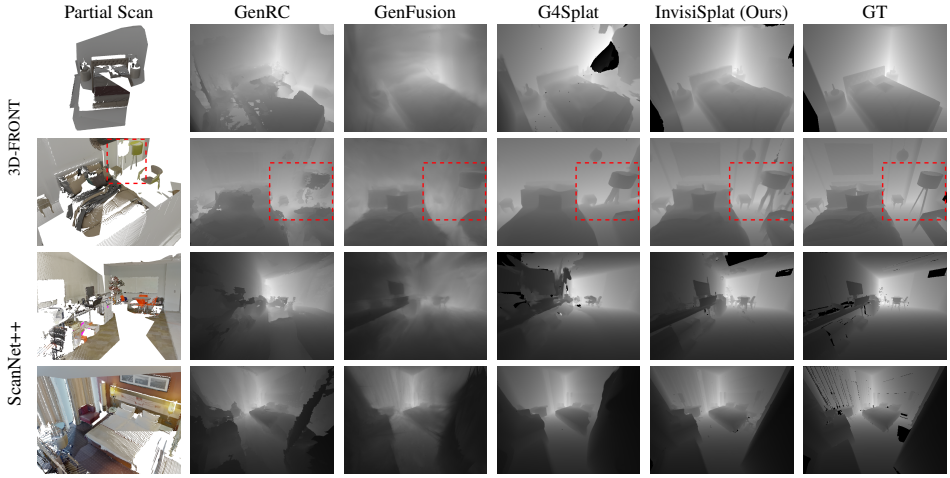

Figure 4: Qualitative comparison of geometry completion.

depth error **by 70%**. Qualitative results in Fig. 4 show that GenRC and G4Splat suffer from large holes and artifacts. GenFusion yields complete but heavily blurred results, likely because Gaussians are stretched to fill missing regions rather than placed on missing surfaces, which would coincide with its high M-OCD. In contrast, our method produces sharp, complete depth maps even for fine-grained objects (e.g., the standing lamp and bedside table in row 2 of Fig. 4). Similar trends hold on real scenes in ScanNet++ (Table 2), where we achieve **76%, 82%, and 42%** relative improvements in CD, M-OCD, and L1 depth error over the best baseline, respectively. Furthermore, our method also demonstrates strong generalization to Replica (Table 3), reducing completion error (M-OCD) **by 93%**, alongside reductions of **74%** in CD and **25%** in L1 depth error over the best baseline.

Appearance quality We present qualitative results of appearance completion in Fig. 5. GenRC's renderings suffer from artifacts due to poor mesh geometry. Those artifacts albeit are usually in small regions, hence do not affect per-pixel metrics such as PSNR and SSIM as much, yet its perceptual metric LPIPS suffers a bit. ViewCrafter produces sharp images but their renderings are not guaranteed to be consistent with the input (e.g., pillows on the bed in row 1 and the originally oval dining table in row 2). GenFusion and G4Splat yield blurred or incomplete results in regions that are not well observed, as their initial 3DGS reconstructions are poor and multi-view inconsistencies from their video diffusion inpainting exacerbate errors. In contrast, our method demonstrates reasonable sharpness in the renderings and produces plausible appearance completion while respecting what has been partially observed. Quantitatively, our method achieves the best PSNR on 3D-FRONT (Table 1), the best LPIPS on ScanNet++ (Table 2), and the best SSIM when generalizing to Replica (Table 3), while ranking second on the remaining appearance metrics for each dataset.

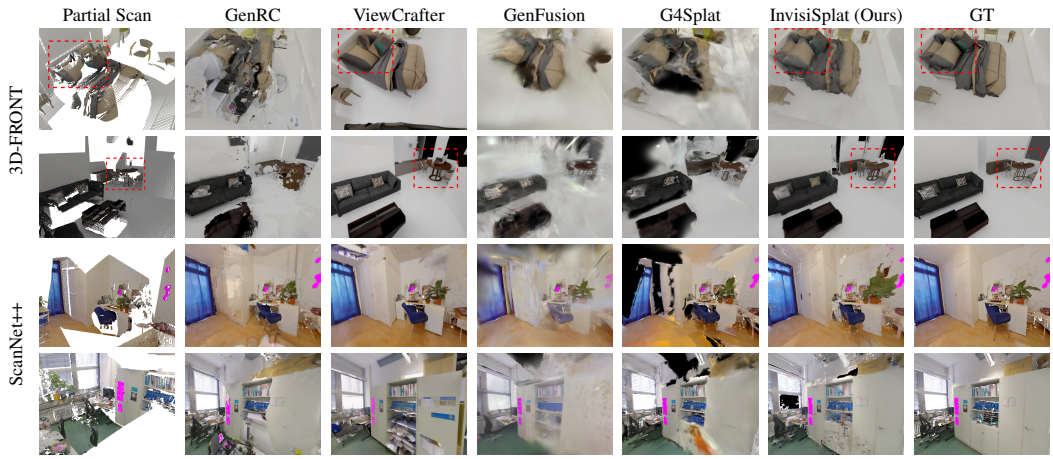

Figure 5: Qualitative comparison of appearance completion.

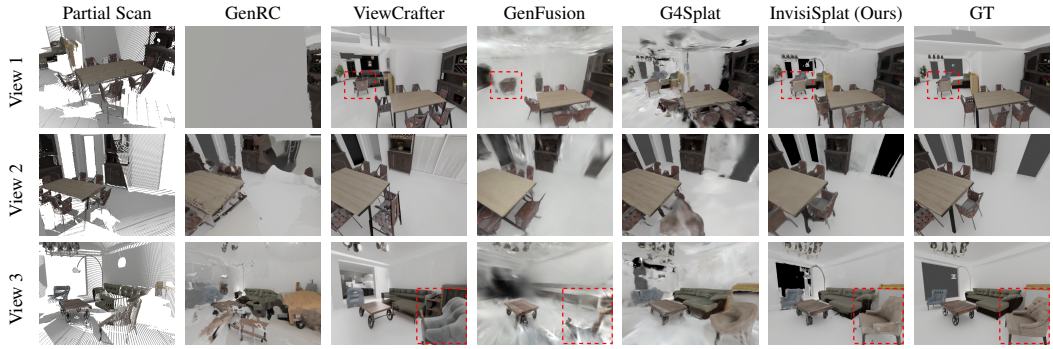

Figure 6: Qualitative comparison of multi-view consistency. ViewCrafter and GenFusion generate chairs inconsistently from different views.

Multi-view consistency Our method achieves strong multi-view consistency by completing both geometry and appearance directly in 3D. In particular, we obtain the best MEt3R score on 3D-FRONT (Table 1) and the second best MEt3R score on ScanNet++ (Table 2) and Replica (Table 3), slightly behind GenRC. GenRC also benefits from strong consistency due to panoramic inpainting, though artifacts in its reconstructed meshes can still produce rendering inconsistencies as shown in Fig. 6. On the other hand, video diffusion-based methods such as ViewCrafter and GenFusion exhibit poor consistency, reflected by their high MEt3R scores. In Fig. 6, we demonstrate this qualitatively, where the geometry and appearance of objects in the background of view 1 change as they come into the foreground in view 3. In contrast, our method maintains visually consistent geometry and appearance across views.

Ablations Table 4 presents ablations on the architecture and training framework of InvisiSplat. Starting from a fully global attention-based flow transformer similar to DiT [67], we incrementally add our proposed components. Introducing local attention improves missing geometry completion (M-OCD) and L1 depth error by 13% and 30%, respectively, while also substantially improving appearance metrics, highlighting the importance of local geometric reasoning for high-fidelity completion and rendering. DINO conditioning provides useful semantic cues, while learning OT flows can produce straighter sampling trajectories, both of which we find improves geometry and appearance. Finally, our weighted flow loss emphasizes fine-grained object geometry over simple structures such as walls, further improving M-OCD by 8%, LPIPS by 9%, and multi-view consistency by 4%.

In Fig. 7, we present a qualitative comparison of our point flow model ablations. While the fully global attention based flow transformer model is able to move point mass to the correct locations, it struggles to form actual surfaces, leading to more complex objects being represented as blobs.

Table 4: Point flow model architecture and training ablations.

	Geometry				Appearance			Consistency
	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
Global Attention	0.0485	0.0004	0.0474	0.1727	16.78	0.6630	0.4713	0.5313
+ Local Attention	0.0463	0.0004	0.0410	0.1200	18.10	0.7115	0.4414	0.5338
+ DINO Conditioning	0.0422	**0.0003**	0.0394	0.1052	18.41	0.7210	0.4269	0.5238
+ OT Flow	**0.0404**	**0.0003**	0.0370	**0.0876**	18.81	0.7363	0.3944	0.5258
+ Weighted Loss	0.0408	**0.0003**	**0.0338**	0.0931	**19.05**	**0.7505**	**0.3581**	**0.5007**

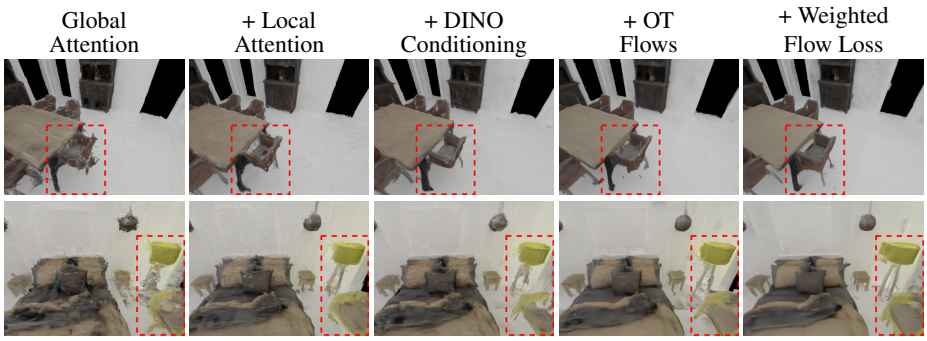

Figure 7: Qualitative comparison of our ablated point flow model.

Adding in local attention helps our flow transformer reason about local structures, which we observe helps form surfaces with fine geometric structure for objects that were originally blob like. Semantic information provided by conditioning the model on DINO features further helps with distinguishing objects that are close to each other, leading to sharper object boundaries, as shown with the chair and table in row 1 of Fig. 7. Furthermore, learning optimal transport (OT) flows can produce straighter sampling trajectory curvatures, which we observe can lead to better generation quality, such as in row 2 where legs of the standing lamp and chair are captured. Finally, adding in our weighted flow loss helps prioritize reconstructing objects and their thin geometric structures. We observe that this improves object sharpness while also helping recover thin structures that were previously missing, such as the back left leg of the chair in row 1 of Fig. 7.

5 Conclusion

We present a novel framework, InvisiSplat, for joint geometry and appearance scene completion from sparse RGB-D observations. The approach consists of two stages, a Gaussian Splat VAE for learning a point cloud encoder and a decoder to predict Gaussian Splat parameters, as well as a second stage that features a two-step flow model that learns to generate samples conditioned on the point cloud latents. Different from other approaches that use implicit tokens or image/text latents, we condition on the point cloud latents which allowed for alternating local 3D attention and global attention. This balanced the needs for precise local geometry and large-scale spatial context and significantly outperformed using only global attention on both geometry and rendering metrics. By performing completion directly in 3D space, our approach naturally handles occlusion, is geometrically consistent across viewpoints, and can respect the partially observed geometry. Across several indoor datasets, we demonstrate that our method produces $33\% - 93\%$ higher fidelity 3D geometry over prior work while attaining competitive performance on appearance metrics when compared to existing approaches.

Currently, our method is limited to indoor scenes due to the lack of available outdoor training data for geometry completion. Existing outdoor datasets primarily support reconstruction and lack the complete ground-truth geometry, including occluded or unobserved regions, needed for this task. Moreover, our approach is limited to completing the missing content in and around the partially observed regions present in the input views. Our method is not suited for extreme viewpoint changes that require complete hallucination of scene content (e.g., when no partially observed content lies within the viewing frustum) nor does it support infinite world generation.

Acknowledgments and Disclosure of Funding

This work was partially supported by NSF grants 1751402 and 2321851, the DARPA TIAMAT grant HR0011-24-9-0423 and AFRL contract FA8750-23-C-0519. The authors acknowledge the National Artificial Intelligence Research Resource (NAIRR) Pilot project NAIRR240433 and an NVIDIA Academic Grant for supporting partially the computing resources.

References

[1] 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.

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[3] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[4] 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 *European Conference on Computer Vision*, pages 370–386. Springer, 2024.

[5] 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 *European Conference on Computer Vision*, pages 1–19. Springer, 2024.

[6] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

[7] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4349–4359, 2025.

[8] Songchun Zhang, Huiyao Xu, Sitong Guo, Zhongwei Xie, Hujun Bao, Weiwei Xu, and Changqing Zou. Spatialcrafter: Unleashing the imagination of video diffusion models for scene reconstruction from limited observations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27794–27805, 2025.

[9] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[10] Chenjie Cao, Chaohui Yu, Shang Liu, Fan Wang, Xiangyang Xue, and Yanwei Fu. Mvgenmaster: Scaling multi-view generation from any image via 3d priors enhanced diffusion model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6045–6056, 2025.

[11] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754, 2017.

[12] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2018.

[13] Jie Li, Yu Liu, Dong Gong, Qinfeng Shi, Xia Yuan, Chunxia Zhao, and Ian Reid. Rgbd based dimensional decomposition residual network for 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7693–7702, 2019.

[14] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 3d sketch-aware semantic scene completion via semi-supervised structure prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4193–4202, 2020.

[15] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12498–12507, 2021.

[16] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5499–5509, 2021.

[17] Haoran Zhou, Yun Cao, Wenqing Chu, Junwei Zhu, Tong Lu, Ying Tai, and Chengjie Wang. Seedformer: Patch seeds based point cloud completion with upsample transformer. In *European conference on computer vision*, pages 416–432. Springer, 2022.

[18] Wesley Khademi and Li Fuxin. Diverse shape completion via style modulated generative adversarial networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] Zhikai Chen, Fuchen Long, Zhaofan Qiu, Ting Yao, Wengang Zhou, Jiebo Luo, and Tao Mei. Anchorformer: Point cloud completion from discriminative nodes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13581–13590, 2023.

[20] Wesley Khademi and Fuxin Li. Point-based instance completion with scene constraints. In *The Thirteenth International Conference on Learning Representations*, 2025.

[21] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5480–5490, 2022.

[22] Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8254–8263, 2023.

[23] Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. Cor-gs: sparse-view 3d gaussian splatting via co-regularization. In *European conference on computer vision*, pages 335–352. Springer, 2024.

[24] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pages 145–163. Springer, 2024.

[25] Hyunwoo Park, Gun Ryu, and Wonjun Kim. Dropgaussian: Structural regularization for sparse-view gaussian splatting. In *Proceedings of the computer vision and pattern recognition conference*, pages 21600–21609, 2025.

[26] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21551–21561, 2024.

[27] Xinhang Liu, Jiaben Chen, Shiu-hong Kao, Yu-Wing Tai, and Chi-Keung Tang. Deceptive-nerf/3dgs: Diffusion-generated pseudo-observations for high-quality sparse-view reconstruction. In *European Conference on Computer Vision*, pages 337–355. Springer, 2024.

[28] Jay Zhangjie Wu, Yuxuan Zhang, Haithem Turki, Xuanchi Ren, Jun Gao, Mike Zheng Shou, Sanja Fidler, Zan Gojcic, and Huan Ling. Difix3d+: Improving 3d reconstructions with single-step diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26024–26035, 2025.

[29] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin Brualla, Pratul Srinivasan, Jonathan Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *Advances in Neural Information Processing Systems*, 37:75468–75494, 2024.

[30] Jensen Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12405–12414, 2025.

[31] Sherwin Bahmani, Ivan Skorokhodov, Aliaksandr Siarohin, Willi Menapace, Guocheng Qian, Michael Vasilkovsky, Hsin-Ying Lee, Chaoyang Wang, Jiaxu Zou, Andrea Tagliasacchi, et al. Vd3d: Taming large video diffusion transformers for 3d camera control. *arXiv preprint arXiv:2407.12781*, 2024.

[32] Sherwin Bahmani, Ivan Skorokhodov, Guocheng Qian, Aliaksandr Siarohin, Willi Menapace, Andrea Tagliasacchi, David B Lindell, and Sergey Tulyakov. Ac3d: Analyzing and improving 3d camera control in video diffusion transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22875–22889, 2025.

[33] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6121–6132, 2025.

[34] Jianhong Bai, Menghan Xia, Xiao Fu, Xintao Wang, Lianrui Mu, Jinwen Cao, Zuozhu Liu, Haoji Hu, Xiang Bai, Pengfei Wan, et al. Recammaster: Camera-controlled generative rendering from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14834–14844, 2025.

[35] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24846–24857, 2025.

[36] Hanwen Liang, Junli Cao, Vidit Goel, Guocheng Qian, Sergei Korolev, Demetri Terzopoulos, Konstantinos N Plataniotis, Sergey Tulyakov, and Jian Ren. Wonderland: Navigating 3d scenes from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 798–810, 2025.

[37] Yuanbo Yang, Jiahao Shao, Xinyang Li, Yujun Shen, Andreas Geiger, and Yiyi Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2857–2869, 2025.

[38] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[39] Stanislaw Szymanowicz, Chrisitian 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.

[40] 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 *European Conference on Computer Vision*, pages 1–20. Springer, 2024.

[41] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024.

[42] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024.

[43] Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. *Advances in Neural Information Processing Systems*, 37:121859–121881, 2024.

[44] Yushi Lan, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. Gaussiananything: Interactive point cloud latent diffusion for 3d generation. In *ICLR*, 2025.

[45] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native diffusion and interactive geometry refiner. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5307–5317, 2025.

[46] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21469–21480, 2025.

[47] Xiaoliang Ju, Zhaoyang Huang, Yijin Li, Guofeng Zhang, Yu Qiao, and Hongsheng Li. Diffindscene: Diffusion-based high-quality 3d indoor scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4526–4535, 2024.

[48] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics (ToG)*, 43(4):1–17, 2024.

[49] Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. Lt3sd: Latent trees for 3d scene diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 650–660, 2025.

[50] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner. L3dg: Latent 3d gaussian diffusion. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[51] Quankai Gao, Iliyan Georgiev, Tuanfeng Y Wang, Krishna Kumar Singh, Ulrich Neumann, and Jae Shin Yoon. Can3tok: Canonical 3d tokenization and latent modeling of scene-level 3d gaussians. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9320–9331, 2025.

[52] Sikuang Li, Chen Yang, Jiemin Fang, Taoran Yi, Jia Lu, Jiazhong Cen, Lingxi Xie, Wei Shen, and Qi Tian. Worldgrow: Generating infinite 3d world. *arXiv preprint arXiv:2510.21682*, 2025.

[53] Yinyu Nie, Ji Hou, Xiaoguang Han, and Matthias Nießner. Rfd-net: Point scene understanding by semantic instance reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4608–4618, 2021.

[54] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Point scene understanding via disentangled instance mesh reconstruction. In *European conference on computer vision*, pages 684–701. Springer, 2022.

[55] Haoang Li, Jinhu Dong, Binghui Wen, Ming Gao, Tianyu Huang, Yun-Hui Liu, and Daniel Cremers. Ddit: Semantic scene completion via deformable deep implicit templates. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21894–21904, 2023.

[56] Yixin Chen, Junfeng Ni, Nan Jiang, Yaowei Zhang, Yixin Zhu, and Siyuan Huang. Single-view 3d scene reconstruction with high-fidelity shape and texture. In *2024 International Conference on 3D Vision (3DV)*, pages 1456–1467. IEEE, 2024.

[57] Andreea Ardelean, Mert Özer, and Bernhard Egger. Gen3dsr: Generalizable 3d scene reconstruction via divide and conquer from a single view. In *2025 International Conference on 3D Vision (3DV)*, pages 616–626. IEEE, 2025.

[58] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[59] Jiabao Lei, Jiapeng Tang, and Kui Jia. Rgbd2: Generative scene synthesis via incremental view inpainting using rgbd diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8422–8434, 2023.

[60] Ming-Feng Li, Yueh-Feng Ku, Hong-Xuan Yen, Chi Liu, Yu-Lun Liu, Albert YC Chen, Cheng-Hao Kuo, and Min Sun. Genrc: Generative 3d room completion from sparse image collections. In *European Conference on Computer Vision*, pages 146–163. Springer, 2024.

[61] Avinash Paliwal, Xilong Zhou, Wei Ye, Jinhui Xiong, Rakesh Ranjan, and Nima Khademi Kalantari. Ri3d: Few-shot gaussian splatting with repair and inpainting diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25094–25103, 2025.

[62] Sibo Wu, Congrong Xu, Binbin Huang, Andreas Geiger, and Anpei Chen. Genfusion: Closing the loop between reconstruction and generation via videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6078–6088, 2025.

[63] Junfeng Ni, Yixin Chen, Zhifei Yang, Yu Liu, Ruijie Lu, Song-Chun Zhu, and Siyuan Huang. G4splat: Geometry-guided gaussian splatting with generative prior. In *The Fourteenth International Conference on Learning Representations*, 2026.

[64] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019.

[65] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[66] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[67] 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.

[68] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

[69] Ka-Hei Hui, Chao Liu, Xiaohui Zeng, Chi-Wing Fu, and Arash Vahdat. Not-so-optimal transport flows for 3d point cloud generation. *arXiv preprint arXiv:2502.12456*, 2025.

[70] Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant flow matching with hybrid probability transport for 3d molecule generation. *Advances in Neural Information Processing Systems*, 36:549–568, 2023.

[71] Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*, 36:59886–59910, 2023.

[72] Mathias Vogel, Keisuke Tateno, Marc Pollefeys, Federico Tombari, Marie-Julie Rakotosaona, and Francis Engelmann. P2p-bridge: Diffusion bridges for 3d point cloud denoising. In *European Conference on Computer Vision*, pages 184–201. Springer, 2024.

[73] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[74] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129(12):3313–3337, 2021.

[75] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[76] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[77] Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6034–6044, 2025.

[78] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[79] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024.

[80] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[81] Baorui Ma, Huachen Gao, Haoge Deng, Zhengxiong Luo, Tiejun Huang, Lulu Tang, and Xinlong Wang. You see it, you got it: Learning 3d creation on pose-free videos at scale. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2016–2029, 2025.

[82] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[83] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pages 71–91. Springer, 2024.

[84] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.

[85] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5294–5306. IEEE, 2025.

A Appendix overview

- Model details (Section B): we provide more details of our model architecture
- Evaluation setup (Section C): we provide more details about our evaluation setup
- Results (Section D): we share more results and ablations of our method

B Model details

B.1 Gaussian Splat VAE

Encoder Our VAE encoder is based on the encoder proposed in [20], consisting of 3 downsampling blocks with PointConv [64] layers for feature interpolation and aggregation. For downsampling point clouds in our first two downsampling blocks, we use grid subsampling with voxel sizes of 0.02 and 0.04 for layers 1 and 2, respectively. In the final downsampling layer, we perform furthest point sampling to sample a fixed point set size of $N_{sparse} = 2048$ points. The final downsampled N_{sparse} points serve as the size of our point cloud latent, and is the point set size that our flow models aim to generate. For PointConv layers, we use k-nearest neighbors for neighborhood sampling and set $k = 16$.

Decoder Our VAE decoder aims to reconstruct a dense point cloud $\hat{\mathbf{P}}_{dense} \in \mathbb{R}^{N_{dense} \times 3}$ with features $\mathbf{F}_3 \in \mathbb{R}^{N_{dense} \times C}$ from a point cloud latent with points $\mathbf{P}_{sparse} \in \mathbb{R}^{N_{sparse} \times 3}$ and features $\mathbf{F}_{latent} \in \mathbb{R}^{N_{sparse} \times d}$. The decoder operates in a coarse-to-fine fashion, consisting of 3 upsampling blocks, where each block upsamples the point cloud by a factor of $r = 2$ to produce the final dense point cloud with $N_{dense} = 16384$ points. For the local attention and Upsample Transformer [17] layer used in our upsampling blocks, we use a neighborhood size of $k = 16$ points. For the global attention in our upsampling blocks, we use multi-head attention with $h = 4$ heads. We set $C = 256$ for the dimension size of the final upsampled features $\mathbf{F}_3$.

Gaussian Prediction Head Our Gaussian Prediction Head aims to generate Gaussian Splat parameters from the final upsampled point and feature set $(\hat{\mathbf{P}}_{dense}, \mathbf{F}_3)$. We choose to use 2DGS [3] as our Gaussian splat parameterization, where each Gaussian $\mathbf{g}_k \in \mathcal{G}$ is parameterized by its central point $\mathbf{p}_k$, two principal tangential vectors $\mathbf{t}_k^u$ and $\mathbf{t}_k^v$ for orientation, a scaling vector $\mathbf{s}_k = (s_k^u, s_k^v)$, an opacity α_k, and view-dependent appearance $\mathbf{c}_k$ parameterized by Spherical Harmonics. Each 2D Gaussian parameter (position, scale, rotation, opacity, and color) is predicted by a separate 2 layer MLP.

B.2 Generative point completion

Flow model Both our point flow and latent flow model use the same local-global flow transformer architecture. Throughout our entire local-global flow transformer, a token dimension size of 768 is used. In our global block, we use multi-head global attention for both self attention and cross attention, where we set $h = 8$ for the number of heads. In our local block, we set the neighborhood size to $k = 16$ points in our PointTransformer layers [68].

Optimal Transport (OT) flows For training our point flow model, we construct Optimal Transport (OT) flows between Gaussian noise distribution and our data distribution. In practice, we compute the approximate optimal transport using the Earth Mover's Distance (EMD) from the Python Optimal Transport (POT) package [82]. Specifically, given a Gaussian noise point cloud $\mathbf{P}_{noise} = [\mathbf{p}_0^1, \ldots, \mathbf{p}_0^{N_{sparse}}]$ and a complete target point cloud $\mathbf{P}_{clean} = [\mathbf{p}_1^1, \ldots, \mathbf{p}_1^{N_{sparse}}]$, we solve the assignment problem:

$$\phi^* = \arg\min_{\phi \in \Phi} \sum_{i=1}^{N_{sparse}} \|\mathbf{p}_1^i - \mathbf{p}_0^{\phi(i)}\|_2 \tag{6}$$

where Φ is the set of all permutations. The optimal assignment ϕ^* is used to reorder the Gaussian noise point cloud as $\tilde{\mathbf{P}}_{noise} = [\mathbf{p}_0^{\phi^*(1)}, \ldots, \mathbf{p}_0^{\phi^*(N_{sparse})}]$. Training samples are then constructed via linear interpolation between $\tilde{\mathbf{P}}_{noise}$ and $\mathbf{P}_{clean}$. At inference, we do not reorder Gaussian noise $\mathbf{P}_{noise}$, and directly run our ODE solver to predict coarse completion $\hat{\mathbf{P}}_{clean}$.

We note that we only apply OT flows to our point flow model and use regular conditional flow matching (CFM) for the latent flow model. Since points have 3D coordinates, CFM can match points with far-away noise samples even when closer ones exist, producing long, intersecting ODE trajectories. OT coupling straightens and shortens these trajectories and reduces variance in the learned vector field, directly benefiting point denoising. In contrast, the latent flow model denoises

features anchored to a fixed point cloud, much like latent diffusion for images denoises features at fixed pixel positions, a setting where CFM is known to perform well. OT could in principle be applied to the latents, but the higher-dimensional feature space yields noisier, less reliable couplings at higher cost, so we use CFM instead.

Weighted flow objective To encourage accurate reconstruction of semantically meaningful structures by our point flow model, we apply a weighting to the conditional flow-matching objective as:

$$\mathcal{L}_{CFM} = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[w(\mathbf{x}_1)\|\mathbf{v}_\theta(\mathbf{x}_t,t) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2\right] \tag{7}$$

where $w(\mathbf{x}_1)$ corresponds to a per sample weighting meant to boost foreground objects and their thin structures. Specifically, we denote a sample $\mathbf{x}_1$ from our complete point cloud target distribution as a point $\mathbf{p}_1^i \in \mathbf{P}_{clean}$ and its corresponding sample weight $w(\mathbf{x}_1)$ as w^i, where:

$$w^i = w^i_{fg} \cdot w^i_{density} \tag{8}$$

$$w^i_{fg} = \begin{cases} \max(1, (N_{bg}/N_{fg})^\alpha) & \text{if } \mathbf{p}_1^i \text{ in foreground} \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

$$w^i_{density} = \left(\frac{1}{|N(\mathbf{p}_1^i)|}\sum_{j\in N(\mathbf{p}_1^i)} \|\mathbf{p}_1^i - \mathbf{p}_1^j\|_2\right)^\beta \tag{10}$$

To compute whether a point is in the foreground or background as well as the total the number of background points N_{bg} and foreground points N_{fg} in eq. (9), we use the associated ground truth instance labels that we have for each target point cloud $\mathbf{P}_{clean}$. For our experiments, we set $\alpha = 0.5$, $\beta = 0.5$, and define a fixed neighborhood size of $|N(\mathbf{p}_1^i)| = 12$ for computing eq. (10).

Classifier Free Guidance (CFG) For both flow models, we utilize Classifier Free Guidance (CFG). During training, we use a drop rate of 0.1 on the point cloud conditions. At inference, we use CFG scales of 4.0 and 2.0 on our point flow model for the 3D-Front and ScanNet++ dataset, respectively. For our latent flow model, we use a CFG scale of 2.5 for both datasets.

Completing entire scenes Due to memory constraints, we train our VAE and flow models on scene chunks rather than entire scenes. In our experiments, we define a scene chunk as a $2m \times 2m \times 2m$ cube. However, we note that a chunk size of $2m \times 2m \times 2m$ is rarely ever the size of the entire scene. To enable completion of entire scenes at inference, we sample anchor points from the partially observed geometry and generate completions of the $2m \times 2m \times 2m$ scene chunk surrounding each anchor point. For anchor point sampling, we perform grid subsampling of the partially observed geometry using a voxel size of $1.0m$.

Consistency at chunk boundaries is encouraged by our conditioning setup: while each chunk generates completions for a $2m \times 2m \times 2m$ region, the conditioning context is drawn from the larger partially observed volume surrounding that same chunk center. This gives each chunk visibility of observed geometry beyond its own generation boundary, directly informing the flow model of structure just outside the region it generates. For large objects spanning multiple chunks, neighboring chunks are conditioned on overlapping real observations of that object, encouraging their completions to agree in the shared region. We note this is not a formal guarantee for consistency across multiple chunks, however, we did not observe this to be a significant practical issue in our qualitative results.

C Evaluation setup

In this section, we discuss more about the dataset, baselines, and metrics used in our evaluation setup.

C.1 Dataset

For training and evaluation, we use a mix of real and synthetic indoor scene data For real scene data, we use the ScanNet++ dataset [73], where each scene consists of RGB images, metric depth maps, metric camera poses, and a densely reconstructed scene mesh. For synthetic data, we use

the 3D-FRONT dataset [74, 75], which consists of textured CAD objects placed in room layout orientations. We additionally process the 3D-FRONT meshes by rendering images and depth maps from random camera poses. For our experiments, we resize all images and depth maps to 480×360.

Access to both datasets was obtained by requesting and gaining approval through their official channels. ScanNet++ licensing falls under their *ScanNet++ Terms of Use* and the 3D-FRONT dataset is licensed under *CC-BY-SA-NC 4.0*.

C.2 Baselines

We note that the input assumptions differ across methods. Our method and GenRC [60] take posed RGB-D images as input, GenFusion [62] and G4Splat [63] take posed RGB images, and ViewCrafter [9] takes unposed RGB images. Consequently, each method constructs the partial input point cloud using different initialization procedures: GenFusion performs structure-from-motion, G4Splat uses MASt3R [83], and ViewCrafter uses DUSt3R [84]. To ensure a fair comparison, we modify each method to use the same input setup as ours. Specifically, we provide posed RGB-D images as input and construct the partial input point cloud by directly backprojecting all input views into the 3D world coordinate system using their poses and depths. This removes the possibility that performance differences arise from weaker geometry initialization (e.g., we observed that DUSt3R and MASt3R produced poor reconstructions on our input views).

C.3 Metrics

We formally define some of the geometry reconstruction metrics used in our evaluation. All metrics are computed on point clouds. To standardize point density across methods, we grid subsample all point clouds to a $2cm$ voxel resolution prior to evaluation. For methods that produce meshes as output, such as GenRC [60] and G4Splat [63], we densely sample $500,000$ points from the mesh surface before performing grid subsampling.

Chamfer Distance We use the symmetric Chamfer distance (CD) to measure overall completion and reconstruction quality. Given two point clouds $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and $\mathbf{Q} \in \mathbb{R}^{M\times 3}$, the symmetric Chamfer distance is defined as:

$$CD(\mathbf{P}, \mathbf{Q}) = \frac{1}{|\mathbf{P}|} \sum_{\mathbf{x}\in\mathbf{P}} \min_{\mathbf{y}\in\mathbf{Q}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \frac{1}{|\mathbf{Q}|} \sum_{\mathbf{y}\in\mathbf{Q}} \min_{\mathbf{x}\in\mathbf{P}} \|\mathbf{x} - \mathbf{y}\|_2^2 \tag{11}$$

Partial Reconstruction Quality (P-OCD) To evaluate how well the partial input was respected in the final reconstruction, we measure the one-sided Chamfer distance (OCD) between the partial geometry that was observed by the input views and the predicted geometry. Given two point clouds $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and $\mathbf{Q} \in \mathbb{R}^{M\times 3}$, the one-sided Chamfer distance is defined as:

$$OCD(\mathbf{P}, \mathbf{Q}) = \frac{1}{|\mathbf{P}|} \sum_{\mathbf{x}\in\mathbf{P}} \min_{\mathbf{y}\in\mathbf{Q}} \|\mathbf{x} - \mathbf{y}\|_2^2 \tag{12}$$

Partial reconstruction quality (P-OCD) is then defined as computing the one-sided Chamfer distance with $\mathbf{P}$ representing the partial input point cloud and $\mathbf{Q}$ representing the predicted complete point cloud.

Missing Completion Quality (M-OCD) While symmetric Chamfer distance measures completion quality, it may be skewed by partial reconstruction quality as well as hallucinated geometry that is plausible but does not actually exist in the ground truth. Because of this, we aim to define a metric that solely measures how well the prediction respects the ground truth missing geometry without being influenced by geometry corresponding to the partial input or by extra hallucinated geometry. To accomplish this, we begin by extracting out the geometry that is missing in the partial input cloud from the ground truth complete point cloud. Specifically, we define $\mathbf{P}_{partial}$ to represent the partial point cloud and $\mathbf{P}_{complete}$ to represent the ground truth complete point cloud. We then define the missing point cloud $\mathbf{P}_{missing}$ as the set of points in $\mathbf{P}_{complete}$ whose nearest neighbor in $\mathbf{P}_{partial}$ is greater than some distance threshold τ:

$$\mathbf{P}_{missing} = \{\mathbf{p} \in \mathbf{P}_{complete} \mid \min_{\mathbf{q}\in\mathbf{P}_{partial}} \|\mathbf{p} - \mathbf{q}\|_2 > \tau\} \tag{13}$$

Table 5: Average inference time for scene completion.

GenRC [60]	ViewCrafter [9]	GenFusion [62]	G4Splat [63]	InvisiSplat (Ours)
7m49s	7m57s	21m45s	24m55s	8m35s

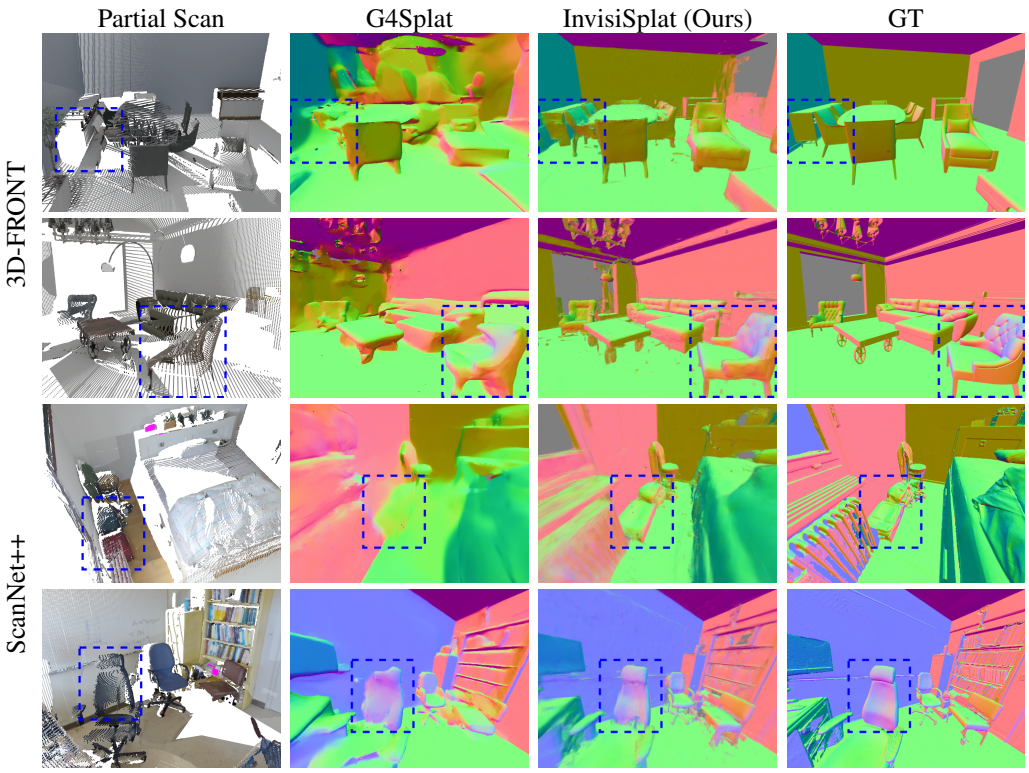

Figure 8: Qualitative comparison of surface normals rendered from reconstructions.

where we set $\tau = 0.02$ (i.e., $2cm$). Missing completion quality (M-OCD) is then defined as computing the one-sided Chamfer distance in eq. (12) with $\mathbf{P}$ representing the missing point cloud $\mathbf{P}_{missing}$ and $\mathbf{Q}$ representing the predicted complete point cloud.

D Results

In this section, we present more quantitative and qualitative results of our method.

D.1 Inference time

We present the average inference time to complete a scene for each method in Table 5. GenFusion [62] and G4Splat [63] suffer from long inference times due to optimizing a 3DGS representation of the scene and performing this optimization several times. ViewCrafter [9] offers much faster inference times but does not produce a final 3D representation, as they only perform RGB video inpainting using a diffusion model. GenRC [60] offers inference times similar to ViewCrafter while also producing a 3D representation of the scene, but the quality of their final completed meshes are poor. Compared to GenFusion and G4Splat, our method offers a $2.5 - 3\times$ improvement in inference speed. While our method is slightly slower than GenRC and ViewCrafter, our approach produces significantly better scene geometry than these methods. Moreover, we note that the inference speed of our method is dependent on the number of scene chunks we need to generate to complete the scene. On average, our scenes consist of 76 chunks per scene.

Table 6: Quantitative results on the 3D-FRONT dataset when using estimated camera poses and depth maps from VGGT.

	Geometry				Appearance		
	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓
GenRC [60]	0.2841	0.0128	0.0841	0.5252	15.06	0.6885	0.4787
ViewCrafter [9]	*	*	*	*	15.52	**0.6886**	**0.4165**
InvisiSplat (Ours)	**0.0451**	**0.0003**	**0.0158**	**0.2958**	**16.06**	0.6572	0.5218

D.2 Geometry

As our method leverages 2DGS [3], we can also render surface normals from our representation. In Fig. 8, we qualitatively compare our reconstructed surface normals to the reconstructed surface normals of G4Splat, which also uses 2DGS. G4Splat can only complete surfaces with simple geometry such as walls and floors, while failing to produce plausible surfaces on more complicated geometry such as the missing parts of the chair and storage cabinet in row 1 of Fig. 8. Moreover, their method struggles to preserve surface geometry that was observed in the partial scan, such as the suitcases in row 3 of Fig. 8. Compared to G4Splat, our method does a better job at preserving the observed surface geometry while producing plausible surface geometry in regions that are missing, even for more complicated objects such as chairs.

D.3 Geometry

In Table 6, we present a comparison of methods when only RGB images are available and the depth maps and camera poses are estimated by VGGT [85]. Even with noisy reconstructed inputs, our method still obtains better geometry completion than GenRC. Moreover, our method achieves the highest PSNR. We note that imperfections in VGGT reconstructions can lead to a singular surface getting represented as multiple surfaces due to backprojection misalignments across views. Our method has not been trained on such data to correct these surface misalignments, which leads to an overall degradation in geometry and appearance quality.

D.4 Ablations

Classifier Free Guidance (CFG) scale In Table 7, we present an ablation on our choice of classifier free guidance scale used by our point flow model for each dataset. For 3D-FRONT, we observe that a guidance scale of 4.0 leads to the best overall performance across geometry, appearance, and consistency. However, on real scans from ScanNet++, we find that a lower guidance scale of 2.0 offers the best performance across all metrics.

Anchor sampling In Table 8, we explore how the choice of voxel size in our anchor sampling impacts the performance of scene completion. We observe that using larger voxel sizes (e.g., 1.5 or 2.0) leads to low performance on our metrics. Using larger voxel sizes for grid subsampling increases the space between sampled anchor points, which can lead to gaps between generated scene chunks where no new geometry is completed. Alternatively, when a voxel size of 1.0 is used for grid subsampling, scene chunks are guaranteed to either have some overlap between each other or have zero overlap but be touching each other (i.e., no gap between scene chunks).

Failure Cases In Fig. 9, we present some failure cases of our method. We observe that our method sometimes fails on objects that have complicated geometry but are not commonly seen in our dataset (e.g., office chair with swivel base or a toilet). This can either lead to noisy and incomplete structures such as in the office chair in row 1 or can lead to noisy surface artifacts that extend out past the object boundaries such as in the toilet case in row 2.

Table 7: Choice of guidance scale parameter for point flow model.

		Geometry				Appearance			Consistency
	Guidance	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
3D-FRONT	1.0	0.0451	0.0004	0.0390	0.1549	17.00	0.6625	0.4803	0.5298
	2.0	**0.0404**	**0.0003**	0.0341	0.1014	18.37	0.7362	0.3785	0.5073
	4.0	0.0408	**0.0003**	**0.0338**	**0.0931**	**19.05**	**0.7505**	**0.3581**	**0.5007**
	6.0	0.0426	**0.0003**	0.0346	0.0978	18.75	0.7433	0.3768	0.5077
ScanNet++	1.0	0.0105	**0.0002**	**0.0090**	0.0791	**16.90**	0.6504	0.4396	0.5100
	2.0	**0.0089**	**0.0002**	**0.0090**	**0.0544**	16.82	**0.6931**	**0.3696**	**0.4959**
	4.0	0.0099	**0.0002**	0.0095	0.0580	16.39	0.6816	0.3856	0.5043
	6.0	0.0107	**0.0002**	0.0102	0.0630	15.94	0.6554	0.4177	0.5167

Table 8: Choice of voxel size (in meters) for grid subsampling of anchor points.

	Geometry				Appearance			Consistency
Voxel Size (in m)	CD ↓	P-OCD ↓	M-OCD ↓	L1 ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MEt3R ↓
1.0	**0.0408**	**0.0003**	**0.0338**	**0.0931**	**19.05**	**0.7505**	**0.3581**	**0.5007**
1.5	0.0492	0.0008	0.0439	0.1198	16.73	0.7285	0.3677	0.5069
2.0	0.0714	0.0139	0.0712	0.5138	10.44	0.5394	0.4622	0.5251

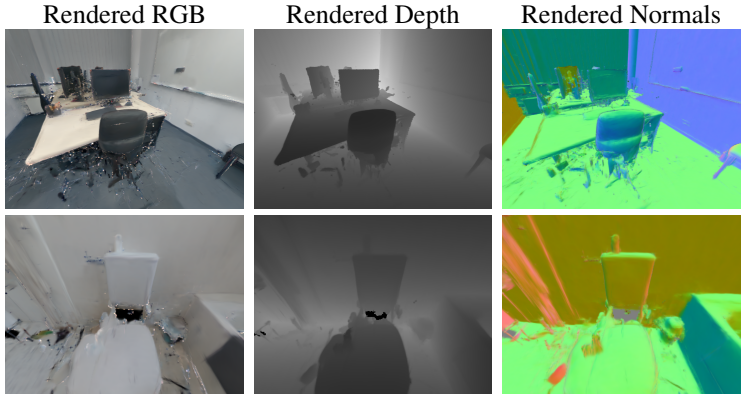

Figure 9: Example failure cases of our method.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The problem scope and our contributions are presented in the abstract. Moreover, in the introduction we further discuss the problem scope and outline our contributions as a bullet pointed list. Claims about our contributions are backed by the experimental results presented in *Section 4.1: Results* of the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the *Conclusion* section of the main paper we discuss some limitations in regards to what types of view extrapolation our method is incapable of handling / is not suited for. In the appendix of our paper, we further present and discuss some failure cases of our method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We do not present any theoretical results in our paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a detailed description of our three models: Gaussian Splat VAE, Point Flow Model, and Structured Latent Flow Model, in *Section 3: Method* of our main paper accompanied by architecture figures in Figures 1, 2, 3. Moreover, we present exact hyperparameters choices of our model architecture in *Section B: Model details* of the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We utilize readily available open source datasets (ScanNet++, 3D-FRONT); however, we have not yet released the code. We plan to release the model and training code in the future with instructions on how to run the model on the datasets experimented with.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: In the *Implementation Details* section of the main paper we provide details of hyperparameters for training. In the *Datasets* section of the main paper we provide the train/test splits of the datasets that we use for training and evaluation. We provide more hyperparameters details related to training/testing in *Section B: Model details* and evaluation settings in *Section C: Evaluation setup* of our appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: In our experiments we do not report error bars or other information about the statistical significance of the experiments. We note that with the resources we have available, the training and inference of our method and the methods we compare against are too computationally expensive to perform multiple training/inference runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In the *Implementation Details* section of the main paper we provide details of the GPUs used for training and the training time of each component of our method. Moreover, we discuss the inference time of our method in *Section D.1: Inference time* of our appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS code of ethics and ensured that the research carried out conforms to it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper proposes a method for completing partially observed scenes which has no noticeable societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not believe the paper poses a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have appropriately cited the datasets (ScanNet++, 3D-FRONT) and open-source models (e.g., DINOv2) used in our paper. Access to datasets were requested and approved through their formal channels, which we mention in the appendix of our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We currently have not released new assets with our paper. In the future, we intend to release model and training code with detailed instructions on how to run the model as well as licensing information.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The proposed work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The proposed work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used in the core methodology and were not involved for the scientific rigor or originality of the research proposed in this work.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.