# RayZer: A Self-supervised Large View Synthesis Model

Hanwen Jiang[1] Hao Tan[2] Peng Wang[2] Haian Jin[3] Yue Zhao[1] Sai Bi[2]
Kai Zhang[2] Fujun Luan[2] Kalyan Sunkavalli[2] Qixing Huang[1] Georgios Pavlakos[1]
[1]The University of Texas at Austin [2]Adobe Research [3]Cornell University

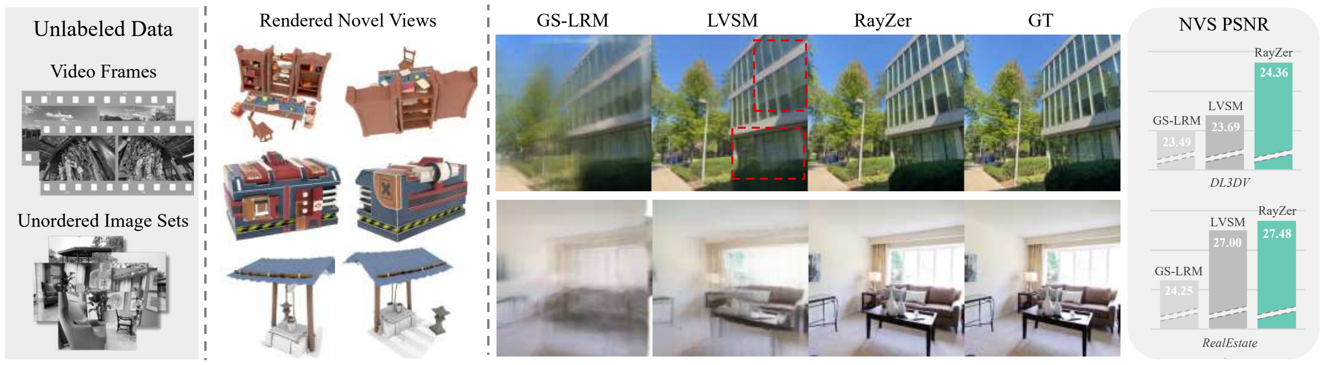

Figure 1. We propose **RayZer**, a **self-supervised** multi-view 3D Vision model trained on *unlabeled data* without any annotations, *e.g.*, camera pose labels. At inference, RayZer supports *feed-forward* novel view synthesis from *unposed & uncalibrated* images. RayZer achieves novel view synthesis performance comparable to that of supervised "oracle" methods (GS-LRM and LVSM), which require camera labels in both training and inference, and even **outperforms** them when they rely on (potentially noisy) COLMAP camera annotations. We show two examples on the right, where COLMAP camera annotations lead to consistent failures of GS-LRM and LVSM during inference.

## Abstract

*We present RayZer, a self-supervised multi-view 3D Vision model trained without any 3D supervision, i.e., camera poses and scene geometry, while exhibiting emerging 3D awareness. Concretely, RayZer takes unposed and uncalibrated images as input, recovers camera parameters, reconstructs a scene representation, and synthesizes novel views. During training, RayZer relies solely on its self-predicted camera poses to render target views, eliminating the need for any ground-truth camera annotations and allowing RayZer to be trained with 2D image supervision. The emerging 3D awareness of RayZer is attributed to two key factors. First, we design a self-supervised framework, which achieves 3D-aware auto-encoding of input images by disentangling camera and scene representations. Second, we design a transformer-based model in which the only 3D prior is the ray structure, connecting camera, pixel, and scene simultaneously. RayZer demonstrates comparable or even superior novel view synthesis performance than "oracle" methods that rely on pose annotations in both training and testing. Project: https://hwjiang1510.github.io/RayZer/*

## 1. Introduction

Self-supervised learning has driven the rise of foundation models, enabling training on vast amounts of unlabeled data and fueled by the scaling law [38]. This paradigm has proven highly effective for LLMs [60], VLMs [2], and visual generation [56]. In contrast, 3D Vision models still rely heavily on ground-truth 3D geometry and camera pose labels [28, 77], which are usually estimated from time-consuming optimization methods, *e.g.*, COLMAP [65], and are not always perfect. This reliance limits both learning scalability and effectiveness. To break free from this constraint, we move beyond the supervised paradigm and ask: *how far can we push a 3D Vision model without any 3D supervision*?

In this paper, we present **RayZer**, a large multi-view 3D model **trained with self-supervision** and **exhibiting emerging 3D awareness**. The input of RayZer is *unposed and uncalibrated* multi-view images, sampled from continuous video frames or unordered multi-view captures. RayZer first recovers the camera parameters, then reconstructs the scene representation, and finally renders novel views. The key insight of our self-supervised training is to use the camera poses *predicted by RayZer itself* to render views that provide *photometric supervision*, rather than following the standard protocol of using ground truth poses for rendering [29, 74, 92]. Thus, RayZer can be trained with **zero 3D supervision**, *i.e.*, no 3D geometry or camera pose supervision. During inference, RayZer predicts camera and scene representations in a *feed-forward* manner, without requiring per-scene optimization. We show inference results in Fig. 1.

As RayZer uses the camera poses predicted by itself for training, this self-supervised task can be interpreted as **3D-aware image auto-encoding** [41, 61, 95]. Initially, RayZer *disentangles* input images into camera parameters and scene representations (reconstruction). It then *re-entangles* these predicted representations back into images (rendering). To facilitate this disentanglement, we **control the information flow**. As shown in Fig. 2, we divide all images into *two parts*: one set predicts the scene representation (input views), while the other offers photometric self-supervision (target views). This is achieved by using estimated poses of the second set to render the scene representation predicted from the first set, thereby preventing trivial solutions that are not 3D-aware.

To facilitate self-supervised learning, RayZer is built only with *transformers* – no 3D representation, hand-crafted rendering equation, or 3D-informed architectures. This design is motivated by self-supervised large models in other modalities [2, 6, 56], enabling RayZer to flexibly and effectively learn domain-specific knowledge. The only 3D prior incorporated in RayZer is the **ray structure**, which simultaneously models the relationship between camera, pixels (image), and scene. Concretely, RayZer first predicts camera poses, which are then converted into pixel-aligned Plücker ray maps [57] to guide the scene reconstruction that follows. This ray-based representation serves as a strong prior for addressing the chicken-and-egg problem of structure and motion [68], effectively allowing the camera and scene representations to regularize each other during training.

We evaluate RayZer on three datasets, including both scene-level and object-level data with different camera configurations. We observe that RayZer demonstrates **comparable or even better novel view synthesis performance** than "oracle" methods [33, 91] that use pose labels in both training and testing. Interestingly, we identify that potentially noisy pose annotations from COLMAP can limit the performance of "oracle" models. The results not only demonstrate the effectiveness of RayZer, but also shows the potential of 3D Vision models to break free from supervised learning.

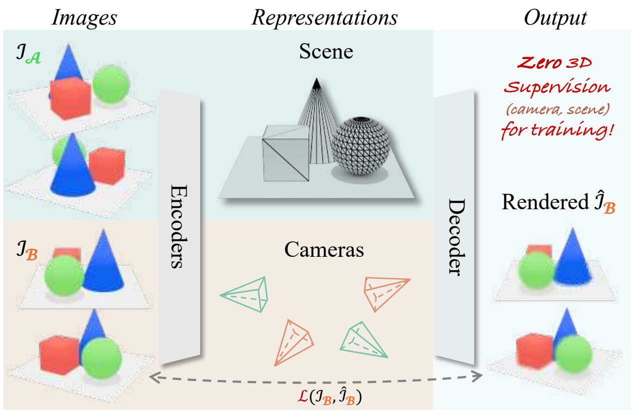
Images
Representations
Output
$\mathcal{I}_\mathcal{A}$
Scene
Zero 3D Supervision
(camera, scene)
for training!
Encoders
Decoder
Rendered $\hat{\mathcal{I}}_\mathcal{B}$
$\mathcal{I}_\mathcal{B}$
Cameras
$\mathcal{L}(\mathcal{I}_\mathcal{B}, \hat{\mathcal{I}}_\mathcal{B})$

Figure 2. **Our proposed self-supervised training framework**. This is an abstract design that we later operationalize with our RayZer model (illustrated in Fig. 3 and Sec. 4). We divide the input images into two sets $\mathcal{I}_\mathcal{A}$ and $\mathcal{I}_\mathcal{B}$. We predict the scene representation from $\mathcal{I}_\mathcal{A}$, and use the predicted cameras of $\mathcal{I}_\mathcal{B}$ (shown in orange) to render the scene. We leverage photometric loss between raw input $\mathcal{I}_\mathcal{B}$ and its prediction $\hat{\mathcal{I}}_\mathcal{B}$ for training.

## 2. Related Work

**Large-scale 3D Vision Models**. 3D Vision models learn 3D representations and priors from data [15, 23, 39, 58, 59, 71, 72, 93, 94]. Recently, researchers have developed large-scale models to acquire general 3D knowledge. One research direction focuses on designing improved model architectures that incorporate the inductive biases of multi-view stereo [10, 14, 75, 86] and epipolar geometry [9, 13, 19, 25]. Another line of work leverages full transformer models that intentionally omit architectural 3D inductive biases [29, 54, 62]. For example, LEAP [29], LRMs [28, 74, 78, 91, 98], and DUSt3R [20, 43, 76, 77, 83] are the first works employing transformers to convert 2D images into 3D representations. SRT [62] and LVSM [33] further replace 3D representations and physical rendering equations with latent representations and learned rendering functions, improving performance and scalability. However, they still require ground-truth camera poses for supervised training and/or accurate camera annotations during inference. To achieve scalable supervised learning, MegaSynth [32] and Stereo4D [34] leverage synthetic data and stereo videos to expand the data scale, however, curating data for different tasks can be laborious. In contrast, RayZer explores self-supervised training to break free from supervised learning.

**Self-supervised 3D Representation Learning**. Learning 3D-aware representations from unlabeled image data is a long-standing problem in 3D Vision. One line of work leverages single-view images. However, they either only work for a specific category [7, 37, 47, 52, 53, 82] or can only recover partial observations [8, 64, 81]. Some works explore semi-supervised learning and achieve better scalability [30, 84], but performance is still highly restricted to the model weights, which are initialized by fully supervised training [85]. The most relevant work is self-supervised learning from multi-view images [69, 79, 80]. For example, Zhou et al. [95], Lai et al. [41], and their following works [21, 87] use camera motion as 2D or 3D warping operations to regularize learning. However, this strong inductive bias limits the learning effectiveness. RUST [63] is a pioneering work in learning latent scene representations from unposed imagery. RayZer is different in three aspects. First, RayZer initially estimates camera poses and uses poses to condition the following latent reconstruction. In contrast, RUST operates in an inverse pipeline – it first reconstructs the scene and then estimates the camera poses. Second, RayZer employs different explicit pose representations to improve information disentanglement and 3D awareness, enabling novel view synthesis by geometrically interpolating predicted poses. Instead, RUST uses a latent pose representation, which makes scene-pose disentanglement challenging and they are not explicitly 3D-aware. Third, RayZer follows the model architecture of LVSM [33] using pure self-attention in transformers, which is different from RUST that follows SRT using convolution and cross-attention.

**Optimization-based Unsupervised SfM, SLAM, and NVS**. Although these methods are not directly comparable to RayZer, we discuss them due to the similar input-output formulations. In detail, these methods optimize target predictions on a per-scene basis [65, 67], while RayZer is a feed-forward parametrized model, learning priors by training on large data. The traditional SfM, SLAM, and NVS methods are unsupervised [26, 65]. Although generally performing well, they are restricted by the complicated hand-crafted workflow, leading to requirements of dense-view inputs [89], slow speed [65], and sensitivity to hyper-parameters [70]. Recent optimization-based NeRF and 3DGS works can also perform NVS from unposed images [4, 22, 48, 66]. However, they do not have learnable model parameters to encode priors, thus requiring off-the-shelf models trained with 3D supervision as regularization or providing initialization.

## 3. Preliminaries

We introduce two important building blocks of RayZer, *i.e.*, the latent set scene representation and how to render it.

**Latent set scene representation**. Compressing data into tokens in latent space is a common practice in text, image, video, etc. Recently, this representation has also been extended to 3D research [33, 62, 63, 88]. In contrast to classical explicit (*e.g.*, meshes and point clouds), implicit (*e.g.*, NeRF [51] and SDF [55]), and hybrid (*e.g.*, triplane [8] and 3DGS [40]) representations that are 3D-aware, the latent set representation is *not explicitly 3D-aware*. It serves as a *compression* of scene information, where the 3D-awareness properties are *fully learned*. The latent set scene representation can be denoted as $\mathbf{z} \in \mathbb{R}^{n \times d}$, where $n$ is the number of tokens in the set and $d$ is the latent feature dimension.

**Rendering latent set scene representation** requires a network, say $R^\theta$, as introduced by SRT [62] and LVSM [33]. We formulate it as $v = R^\theta(\mathbf{z}, r)$, where $r$ is a ray and $v$ is the rendered property, *e.g.*, RGB values, of the corresponding pixel[1]. This formulation is the same as traditional Graphics rendering techniques [1, 36], as $v = R(\text{SCENE}, \text{RAY})$, where $R$ is a pre-defined and handcrafted rendering equation, *e.g.*, alpha-blending ray marching in NeRF. Differently, our "rendering equation" is a learned model parameterized with weights $\theta$, and our scene representation is a latent token set as discussed previously. We omit the model parametrization, *e.g.*, weight $\theta$, in the following description for clarity.

[1]For improved efficiency and performance, LVSM groups rays from the same image patch and decodes them jointly.

## 4. RayZer

In this section, we first introduce RayZer's self-supervised learning framework (Sec. 4.1). Then, we present the details of the RayZer model architecture (Sec. 4.2).

### 4.1. RayZer's Self-supervised Learning

We first formulate the input and output of RayZer. We then introduce the self-supervised learning framework.

We focus on the standard setting of modeling static scenes [65]. The **input** of RayZer is a set of *unposed and uncalibrated* multi-view images $\mathcal{I} = \{I_i \in \mathbb{R}^{H \times W \times 3} | i = 1, ..., K\}$, which can come from unlabeled video frames or image sets. The **output** is a parametrization of the inputs, *i.e.*, camera intrinsics, per-view camera poses, and scene representation, enabling novel view synthesis. To predict these representations, we build the RayZer model and train it with *self-supervised learning* – no 3D supervision, *i.e.*, 3D geometry, and camera pose annotations during training.

To train RayZer with self-supervision, we control the data information flow. We split the input images $\mathcal{I}$ into **two non-overlapping subsets** $\mathcal{I}_\mathcal{A}$ and $\mathcal{I}_\mathcal{B}$, where $\mathcal{I}_\mathcal{A} \cup \mathcal{I}_\mathcal{B} = \mathcal{I}$ and $\mathcal{I}_\mathcal{A} \cap \mathcal{I}_\mathcal{B} = \emptyset$. RayZer uses $\mathcal{I}_\mathcal{A}$ to predict the *scene representation*, and use $\mathcal{I}_\mathcal{B}$ *for providing supervision*. Thus, RayZer renders images that correspond to $\mathcal{I}_\mathcal{B}$, denoted as $\hat{\mathcal{I}}_\mathcal{B}$, and we apply photometric losses:

$$\mathcal{L} = \frac{1}{K_\mathcal{B}} \sum_{\hat{I} \in \hat{\mathcal{I}}_\mathcal{B}} (\texttt{MSE}(I, \hat{I}) + \lambda \cdot \texttt{Percep}(I, \hat{I})), \quad (1)$$

where $K_\mathcal{B} = |\mathcal{I}_\mathcal{B}|$ is the size (number of images) of $\mathcal{I}_\mathcal{B}$ , $I \in \mathcal{I}_\mathcal{B}$ is the image that corresponds to a predicted image $\hat{I}$, and $\lambda$ is the weight for perceptual loss [35, 46]. The two sets are randomly sampled during training.

### 4.2. RayZer Model

**Overview**. As introduced in Sec. 4.1, RayZer recovers both camera parameters and the scene representation from unposed, uncalibrated input images. A key design element of RayZer is its **cascaded** prediction of camera and scene representations. This is motivated by the fact that even noisy cameras can be a strong condition for better scene reconstruction [31, 65, 92], which is analogous to traditional structure-from-motion methods [65] and is in contrast with recent reconstruction-first methods [63, 74, 77]. This design can provide mutual regularization of predicting pose and scene during training, facilitating self-supervised learning.

RayZer builds a pure transformer-based model, benefiting from its scalability and flexibility. As shown in Fig. 3, RayZer first tokenizes input images and uses a transformer-based encoder to predict camera parameters of *all views*. In this step, the cameras are represented by their intrinsics and `SE(3)` camera poses. This **low-dimensional**, **geometri-**

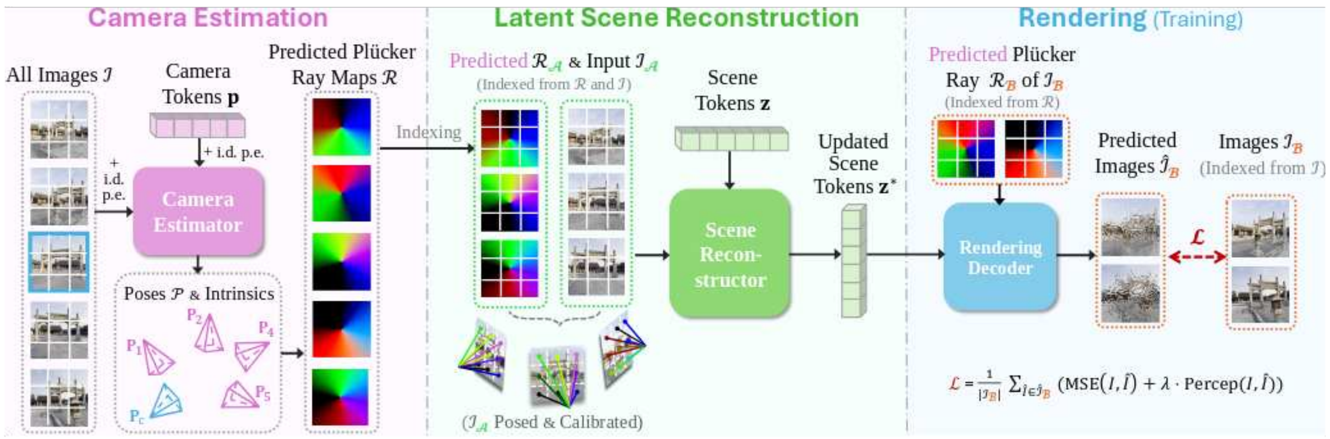
Camera Estimation
Latent Scene Reconstruction
Rendering (Training)
All Images $\mathcal{I}$
Camera Tokens $\mathbf{p}$
+ i.d. p.e.
+ i.d. p.e.
Camera Estimator
Poses $\mathcal{P}$ & Intrinsics
$P_1$
$P_2$
$P_4$
$P_5$
$P_c$
Predicted Plücker Ray Maps $\mathcal{R}$
Indexing
Predicted $\mathcal{R}_{\mathcal{A}}$ & Input $\mathcal{I}_{\mathcal{A}}$
(Indexed from $\mathcal{R}$ and $\mathcal{I}$)
($\mathcal{I}_{\mathcal{A}}$ Posed & Calibrated)
Scene Tokens $\mathbf{z}$
Scene Recon-structor
Updated Scene Tokens $\mathbf{z}^*$
Predicted Plücker Ray $\mathcal{R}_{\mathcal{B}}$ of $\mathcal{I}_{\mathcal{B}}$
(Indexed from $\mathcal{R}$)
Rendering Decoder
Predicted Images $\hat{\mathcal{I}}_{\mathcal{B}}$
Images $\mathcal{I}_{\mathcal{B}}$
(Indexed from $\mathcal{I}$)
$\mathcal{L}$
$\mathcal{L} = \frac{1}{|\mathcal{I}_{\mathcal{B}}|} \sum_{I \in \mathcal{I}_{\mathcal{B}}} (\text{MSE}(I, \hat{I}) + \lambda \cdot \text{Percep}(I, \hat{I}))$

Figure 3. **RayZer self-supervised learning framework.** RayZer takes in unposed and uncalibrated multi-view images $\mathcal{I}$ and predicts per-view camera parameters and a scene representation, which supports novel view rendering. (**Left**) RayZer first estimates camera parameters, where one view is selected as the canonical reference view (in blue box). RayZer predicts the intrinsics and the relative camera poses $\mathcal{P}$ of all views. The predicted cameras are then converted into pixel-aligned Plücker ray maps $\mathcal{R}$. (**Middle**) RayZer uses a subset of input images, $\mathcal{I}_{\mathcal{A}}$, as well as their previously predicted camera Plücker ray maps, $\mathcal{R}_{\mathcal{A}}$, to predict a latent scene representation. Here, the Plücker ray maps, $\mathcal{R}_{\mathcal{A}}$, serve as an effective condition for scene reconstruction. (**Right**) RayZer can render a target image given the scene representation $\mathbf{z}^*$ and a target camera. During training, we use $\mathcal{R}_{\mathcal{B}}$, which is the previously predicted cameras Plücker ray maps of $\mathcal{I}_{\mathcal{B}}$, to render $\hat{\mathcal{I}}_{\mathcal{B}}$. This allows training RayZer end-to-end with self-supervised photometric losses between inputs $\mathcal{I}_{\mathcal{B}}$ and their renderings $\hat{\mathcal{I}}_{\mathcal{B}}$.

**cally well-defined** parametrization helps disentangle image information from the camera representation.

RayZer then transforms the `SE(3)` camera poses and intrinsics into Plücker ray maps [57], representing the predicted cameras as pixel-aligned rays. This ray-based representation captures both the 2D ray-pixel alignment and the 3D ray geometry, providing **fine-grained**, ray-level details that encapsulate the physical properties of the camera model. The ray maps serve as a condition for improving the reconstruction stage that follows.

From the image and predicted Plücker rays of $\mathcal{I}_{\mathcal{A}}$, RayZer uses another transformer-based encoder to predict the latent set scene representation (introduced in Sec. 3 and detailed later). Then, RayZer uses the previously estimated cameras of $\mathcal{I}_{\mathcal{B}}$ to predict $\hat{\mathcal{I}}_{\mathcal{B}}$, providing photometric self-supervision (Eq. 1). We now formally introduce the RayZer model.

**Image Tokenization**. For all $K$ input images $\mathcal{I} = \{I_i \in \mathbb{R}^{H\times W\times 3} | i = 1, ..., K\}$, we patchify them into non-overlapping patches following ViT [18]. Each patch is in $\mathbb{R}^{s\times s\times 3}$, where $s$ is the patch size. We use a linear layer to encode each patch into a token in $\mathbb{R}^d$, leading to a patch-aligned token map $f_i \in \mathbb{R}^{h\times w\times d}$ for each image, where $h = H/s$, $w = W/s$, and $d$ is the latent dimension.

We then add positional embeddings (p.e.) to the tokens, enabling the following model to be aware of the spatial location and the corresponding image index of each token. Specifically, we combine the sinusoidal spatial p.e. [18] and the sinusoidal image index p.e. [3] using a linear layer; note that the image index p.e. is shared among all tokens from the same image. When training on continuous video frames, these image index embeddings also encode sequential priors, which benefits pose estimation. Finally, we reshape the token maps of all images into a set, denoted as $\mathbf{f} \in \mathbb{R}^{Khw\times d}$ (recall that the transformer is invariant to the permutation of tokens). For brevity, we will use this notation for latent token sets throughout the rest of the paper.

**Camera Estimator**. The camera estimator $\mathcal{E}_{cam}$ predicts camera parameters, *i.e.*, camera poses and intrinsics, for all input images. We use a learnable camera token in $\mathbb{R}^{1\times d}$ as the initial feature for this prediction for all views. We repeat the token $K$ times and add them with image index p.e. such that they correspond to the $K$ images. We denote this camera feature initialization as $\mathbf{p} \in \mathbb{R}^{K\times d}$. We then use the camera estimator composed of full self-attention transformer layers to update the camera tokens, as:

$$\{\mathbf{f}^*, \mathbf{p}^*\} = \mathcal{E}_{cam}(\{\mathbf{f}, \mathbf{p}\}), \tag{2}$$

where $\{\cdot,\cdot\}$ denotes concatenation along the token dimension (the union set of two token sets), and $\mathbf{f}^*$ and $\mathbf{p}^*$ are the updated tokens. We note that $\mathbf{f}^*$ is not used for the following computation – it is only used as context to update $\mathbf{p}$ in the transformer layers. For clarity, we formulate the transformer layers as follows:

$$\mathbf{y}^0 = \{\mathbf{f}, \mathbf{p}\}, \tag{3}$$
$$\mathbf{y}^l = \text{TransformerLayer}^l(\mathbf{y}^{l-1}),\ l = 1, ..., l_T \tag{4}$$
$$\{\mathbf{f}^*, \mathbf{p}^*\} = \text{split}(\mathbf{y}^{l_T}), \tag{5}$$

where $l_T$ is the number of layers, and the split operation recovers the two token sets, inverting Eq. 3. This notation remains consistent throughout the rest of the paper.

We then predict the camera parameters for each image independently. For camera pose prediction, we follow prior works of using relative camera poses to resolve ambiguity [31, 89]. We select one view as the canonical reference (*e.g.*, with identity rotation and zero translation), while for every non-canonical view, we predict its relative pose with respect to the canonical view. We parametrize the `SO(3)` rotation using a continuous 6D representation [97], and we predict the relative pose with a two-layer MLP as follows:

$$p_i = \text{MLP}_{pose}([\mathbf{p}_i^*, \mathbf{p}_c^*]), \tag{6}$$

where $[\cdot, \cdot]$ denotes concatenation along the feature dimension, $\mathbf{p}_i^*$ and $\mathbf{p}_c^*$ (all in $\mathbb{R}^d$) are the camera tokens for image $I_i$ and the canonical view, respectively. The output $p_i \in \mathbb{R}^9$ represents the predicted pose parameters, which are then transformed into an `SE(3)` pose $\mathbf{P}_i$ for image $I_i$.

For intrinsics prediction, following prior works [24, 41], we parameterize intrinsics using a single focal length value, under the assumptions that i) the focal lengths along the x and y axes are identical, ii) all views share the same intrinsics, and iii) the principal point is at the image center. We predict the focal length using a two-layer MLP:

$$\text{focal} = \text{MLP}_{focal}(\mathbf{p}_c^*). \tag{7}$$

The predicted focal length is then converted into the intrinsics matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$.

**Scene Reconstructor**. As discussed in Sec. 4.1, we predict the scene representation from image set $\mathcal{I}_\mathcal{A}$ and additionally condition it on the previously predicted camera parameters $\mathcal{P}_\mathcal{A} = \{(\mathbf{P}_i, \mathbf{K}) | I_i \in \mathcal{I}_\mathcal{A}\}$. We first convert $\mathcal{P}_\mathcal{A}$ to pixel-aligned Plücker rays [57] for each image, denoted as $\mathcal{R} \in \mathbb{R}^{K\times H\times W\times 6}$. Similar to image inputs, we also tokenize the Plücker rays into patch-level tokens using a linear layer, yielding $\mathbf{r} \in \mathbb{R}^{Khw\times d}$. We index the image and Plücker rays tokens corresponding to the image set $\mathcal{I}_\mathcal{A}$, denoted as $\mathbf{f}_\mathcal{A}$ and $\mathbf{r}_\mathcal{A}$ (each in $\mathbb{R}^{K_\mathcal{A}hw\times d}$, respectively). We fuse these tokens along the feature dimension with a two-layer MLP:

$$\mathbf{x}_\mathcal{A} = \text{MLP}_{fuse}([\mathbf{f}_\mathcal{A}, \mathbf{r}_\mathcal{A}]), \tag{8}$$

where $\mathbf{x}_\mathcal{A} \in \mathbb{R}^{K_\mathcal{A}hw\times d}$ represents the fused tokens. Importantly, we use the raw image tokens $\mathbf{f}$ rather than the pose transformer output $\mathbf{f}^*$ for this fusion. This design choice prevents leakage of information from the image set $\mathcal{I}_\mathcal{B}$, since the camera estimator transformer producing $\mathbf{f}^*$ has access to a global context that includes tokens from $\mathcal{I}_\mathcal{B}$.

We then employ a scene reconstructor $\mathcal{E}_{scene}$ consisting of full self-attention transformer layers to predict the latent scene representation. To initialize this representation, we use a set of learnable tokens $\mathbf{z} \in \mathbb{R}^{L\times d}$, where $L$ denotes the number of tokens. We formulate the process as follows:

$$\{\mathbf{z}^*, \mathbf{x}_\mathcal{A}^*\} = \mathcal{E}_{scene}(\{\mathbf{z}, \mathbf{x}_\mathcal{A}\}). \tag{9}$$

The update rule is identical to the transformer layers in the camera estimator $\mathcal{E}_{cam}$. Here, $\mathbf{z}^*$ represents the final latent scene representation predicted from $\mathcal{I}_\mathcal{A}$. Meanwhile, $\mathbf{x}_\mathcal{A}^*$ is discarded.

**Rendering Decoder**. We first define the rendering decoder and then describe its training usage.

We use a transformer-based decoder with full self-attention for rendering, following LVSM [33]. For a target image, we begin by representing it as pixel-aligned Plücker rays and tokenize these rays using a linear layer to obtain target tokens $\mathbf{r} \in \mathbb{R}^{hw\times d}$. Next, we fuse the scene information by updating the tokens with a decoder $\mathcal{D}_{render}$ comprising transformer layers:

$$\{\mathbf{r}^*, \mathbf{z}'\} = \mathcal{D}_{render}(\{\mathbf{r}, \mathbf{z}^*\}), \tag{10}$$

where $\mathbf{z}'$ is subsequently discarded, while the update rule of $\mathcal{D}_{render}$ is the same as previously introduced modules. Finally, we decode the RGB values at the patch level with an MLP:

$$\hat{I} = \text{MLP}_{rgb}(\mathbf{r}^*), \tag{11}$$

where $\hat{I} \in \mathbb{R}^{hw\times(3s^2)}$. We reshape $\hat{I}$ to recover the 2D spatial structure, yielding a final rendered image in $\mathbb{R}^{H\times W\times 3}$.

During training, we use the predicted Plücker ray maps $\mathcal{R}_\mathcal{B}$, which correspond to $\hat{\mathcal{I}}_B$, to render images of $\hat{\mathcal{I}}_B$ and then compute the self-supervised loss as defined in Eq. 1.

## 5. Experiments

In this section, we introduce the experimental setting and present the evaluation results. For the implementation, RayZer employs 24 transformer layers, with 8 layers for each of the camera estimator, scene encoder, and rendering decoder. We train RayZer with a learning rate of $4 \times 10^{-4}$ with a cosine scheduler for 50,000 iterations and a batch size of 256. The weight of perceptual loss is $\lambda = 0.2$. For all experiments, we used a resolution of 256 with a patch size of 16. More details are in the Appendix.

### 5.1. Experimental Setup

We introduce our experimental setup, including datasets, evaluation protocol and metrics, as well as baseline methods.

**Datasets**. We use three datasets to evaluate RayZer, including two scene-level datasets, DL3DV [49] and RealEstate [96], and an object-level dataset Objaverse [17] (rendered as videos). We train and test on each dataset separately. The numbers of input views ($\mathcal{I}_\mathcal{A}$) and target views ($\mathcal{I}_\mathcal{B}$) are set to 16 and 8 for DL3DV, 5 and 5 for RealEstate, and 12 and 8 for Objaverse, respectively. We sample input images with the index ranges of 64-96, 128-192, and 50-65 on DL3DV, RealEstate, and Objaverse, respectively. These values are chosen based on data difficulty, especially camera baseline, following prior works [9, 91, 98]. We use the offical DL3DV train-test split, and split RealEstate following [9]. More details can be found in the Appendix.

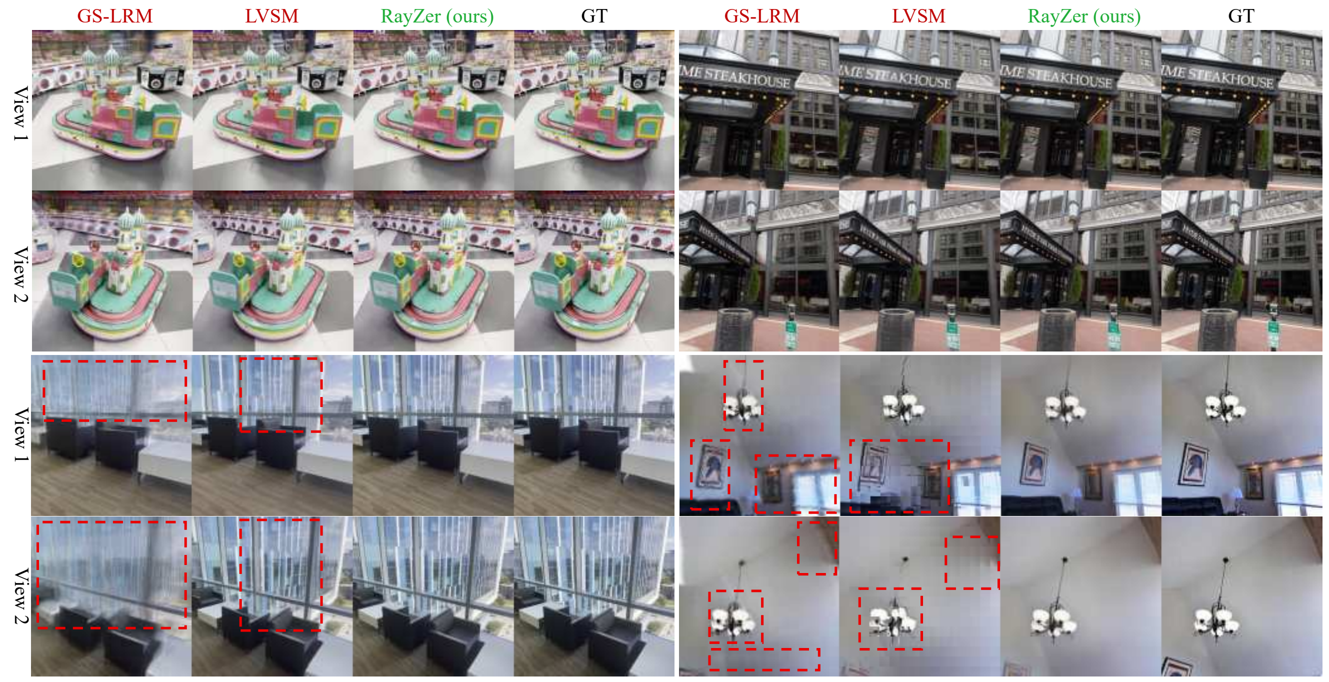

Figure 4. **Visualization results on RealEstate and DL3DV**. We compare RayZer with “oracle” methods GS-LRM and LVSM, which use COLMAP pose annotations in both training and testing. Our self-supervised RayZer model does not use any pose annotations. Generally, RayZer performs on par with “oracle” methods (first row), and can outperform them on cases that COLMAP usually struggles to handle, *e.g.*, glasses and white walls (highlighted with red boxes). The results verify our analysis on the problems of using COLMAP in Sec. 5.2.

**Evaluation Protocol and Metrics**. We evaluate novel view synthesis quality. Specifically, the evaluation protocol of RayZer is different from the “oracle” and supervised methods, which use ground-truth poses to render images. Instead, we use **predicted poses** to render novel views, thereby assessing the compatibility between the predicted poses and the scene representation. Since the model is trained without explicit pose annotations, the learned poses exist in a different space, and their direct correspondence to standard pose annotations is unknown. This evaluation protocol follows RUST [63]. We note that the target views are used only for pose estimation and not for scene representation prediction, ensuring that no information leakage occurs.

**Baselines**. We compare RayZer with two types of methods, including 1) **“oracle” methods**, *i.e.*, GS-LRM [91] and LVSM [33] (encoder-decoder version), that use ground-truth camera poses during **both** training (as supervision) and inference (as pre-requisite). LVSM also uses latent set scene representation. Thus, it serves as the main comparison for the “oracle” methods; 2) **supervised method**, *i.e.*, PF-LRM [74], which requires camera supervision to learn pose estimation and reconstruction; thus, it is pose-free during inference. For fair comparisons, we use 16 transformer layers in total for GS-LRM and LVSM. Thus, their number of parameters is the same as RayZer, except that RayZer has another camera estimator to handle unposed images. We use 24 transformer layers for PF-LRM. We also consider the self-supervised method RUST [63], but since it does not have an official public implementation, we ablate the key design differences between RUST and RayZer in Table 7 instead.

| | Training Supervision | Inference w. COLMAP Cam. | Even Sample PSNR$_\uparrow$ | Even Sample SSIM$_\uparrow$ | Even Sample LPIPS$_\downarrow$ | Random Sample PSNR$_\uparrow$ | Random Sample SSIM$_\uparrow$ | Random Sample LPIPS$_\downarrow$ |
|---|---|---|---|---|---|---|---|---|
| **“Oracle” methods** (assume inputs are posed & use pose annotations during training) | | | | | | | | |
| GS-LRM | 2D + Camera | Yes | 23.49 | 0.712 | 0.252 | 23.02 | 0.705 | 0.266 |
| LVSM | 2D + Camera | Yes | 23.69 | 0.723 | 0.242 | 23.10 | 0.703 | 0.257 |
| **Unsupervised methods** (inputs are un-posed & no pose annotations used during training) | | | | | | | | |
| RayZer | 2D | No | **24.36** | **0.757** | **0.209** | **23.72** | **0.733** | **0.222** |

Table 1. **Evaluation results on DL3DV**. The camera annotations used by the “oracle” models come from **COLMAP**. The results are reported with continuous video frames (ordered) as the input. The results for the unordered image set input are in Table. 4. The input and target views can be evenly or randomly sampled from video frames. We bold our result if it is better than the “oracle” models.

## 5.2. Results

**Main results**. Table 1-3 summarizes the results on the three datasets. Remarkably, without any 3D labels (e.g., camera pose annotations) during training, RayZer achieves performance comparable to the best “oracle” model, LVSM. In fact, RayZer even outperforms LVSM on DL3DV and RealEstate10k while performing slightly worse on Objaverse. We conjecture that this is because the camera poses in DL3DV and RealEstate are annotated by COLMAP, which can be imperfect and set an upper bound for “oracle” methods that are supervised by COLMAP annotations. In contrast, our self-supervised approach enables the model to learn a

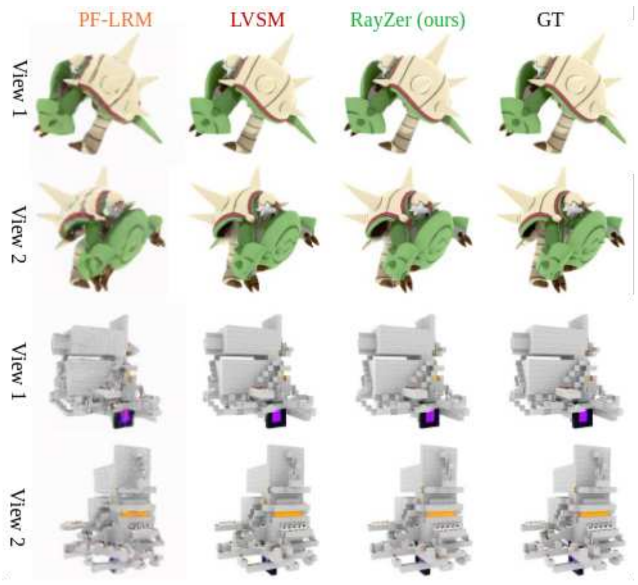

Figure 5. **Visualization results on Objaverse**. RayZer performs on par with LVSM and outperforms the supervised method PF-LRM.

| | Training Supervision | Inference w. COLMAP Cam. | Even Sample PSNR$_\uparrow$ | Even Sample SSIM$_\uparrow$ | Even Sample LPIPS$_\downarrow$ | Random Sample PSNR$_\uparrow$ | Random Sample SSIM$_\uparrow$ | Random Sample LPIPS$_\downarrow$ |
|---|---|---|---|---|---|---|---|---|
| **“Oracle” methods** (assume inputs are posed & use pose annotations during training) | | | | | | | | |
| GS-LRM | 2D + Camera | Yes | 24.25 | 0.770 | 0.227 | 23.21 | 0.748 | 0.251 |
| LVSM | 2D + Camera | Yes | 27.00 | 0.851 | 0.157 | 25.88 | 0.828 | 0.175 |
| **Unsupervised methods** (inputs are un-posed & no pose annotations used during training) | | | | | | | | |
| RayZer | 2D | No | **27.48** | **0.861** | **0.146** | **26.32** | **0.835** | **0.164** |

Table 2. **Evaluation results on RealEstate** with continuous video frames inputs. The camera annotations come from **COLMAP**.

pose space that optimally benefits latent reconstruction and novel view synthesis. This hypothesis is further supported by the results on Objaverse – a synthetic dataset with perfect pose annotations from the rendering tool – where LVSM, acting as a true oracle, outperforms RayZer. Nonetheless, the small performance gap showcases the effectiveness of our self-supervised training. Visualizations in Fig. 4 further support our conjecture regarding COLMAP's noisy poses, as both LVSM and GS-LRM consistently underperform on challenging cases that COLMAP usually fails. These results not only validate our self-supervised learning approach but also demonstrate its potential to break free from the limitations of supervised learning.

**Using unordered image sets for training**. RayZer can be trained on continuous video frames (Table 1-3) or unordered image sets (Table 4). Note that these two training settings are applied separately. As shown in Table 4, we observe that the model trained with unordered image sets performs worse than the one trained with continuous video frames. We notice that the difference is at the pose estimation stage – specifically, the image index positional embedding encourages local pose smoothness that benefits the learning of pose estimation on continuous frames. This finding suggests that scaling training data using video resources, which are plentiful online, could be more advantageous than relying on unordered image sets that are often limited in scale and contain noisy content [45, 73].

| | Training Supervision | Inference w. GT Cam. | Even Sample PSNR$_\uparrow$ | Even Sample SSIM$_\uparrow$ | Even Sample LPIPS$_\downarrow$ | Random Sample PSNR$_\uparrow$ | Random Sample SSIM$_\uparrow$ | Random Sample LPIPS$_\downarrow$ |
|---|---|---|---|---|---|---|---|---|
| **“Oracle” methods** (assume inputs are posed & use pose annotations during training) | | | | | | | | |
| LVSM | 2D + GT Cam. | Yes | 32.34 | 0.950 | 0.050 | 32.34 | 0.949 | 0.051 |
| **Supervised methods** (inputs are un-posed & use pose annotations during training) | | | | | | | | |
| PF-LRM | 2D + GT Cam. | Yes (render) | 25.48 | 0.882 | 0.110 | 25.43 | 0.881 | 0.111 |
| **Unsupervised methods** (inputs are un-posed & no pose annotations used during training) | | | | | | | | |
| RayZer | 2D | No | 31.52 | 0.945 | 0.052 | 31.42 | 0.943 | 0.053 |

Table 3. **Evaluation results on Objaverse** with continuous video frames inputs. The camera annotations are Blender ground-truth. PF-LRM uses ground-truth poses to render novel views, same with oracle methods, and we evaluate its predicted pose in Table 5.

| | Training Supervision | Inf. w. GT Pose | Continuous Inputs | Even Sample PSNR$_\uparrow$ | Even Sample SSIM$_\uparrow$ | Even Sample LPIPS$_\downarrow$ | Random Sample PSNR$_\uparrow$ | Random Sample SSIM$_\uparrow$ | Random Sample LPIPS$_\downarrow$ |
|---|---|---|---|---|---|---|---|---|---|
| (1) | 2D | No | ✓ | 24.36 | 0.757 | 0.209 | 23.72 | 0.733 | 0.222 |
| (2) | 2D | No | ✗ | 20.56 | 0.576 | 0.334 | 20.02 | 0.566 | 0.356 |

Table 4. **Evaluating RayZer performance when using continuous or unordered images for training** on DL3DV. In evaluations, the input frames are sampled from continuous video frames. (1) keeps their temporal continuity (encoded by the image index p.e.) during training. (2) randomly shuffles the images during training.

## 5.3. Analysis of Camera Poses

**RayZer's learned camera pose space**. We visualize some camera poses predicted by RayZer in Fig. 6. Although RayZer predicts `SE(3)` camera poses, we observe that these poses do not exactly match the real-world pose space. This result indicates that the `SE(3)` poses, which are later converted into Plücker ray maps, offer a degree of flexibility. Since both the rendering decoder and the scene representation operate in latent space, RayZer remains robust to any warping between the learned pose space and the actual real-world poses, as long as the poses are compatible with the scene representation and the decoder.

**3D Awareness of predicted camera poses**. We further investigate whether the pose space learned by RayZer is 3D aware. To this end, we interpolate the predicted poses of input views to synthesize more novel views, where the camera pose of a novel view is interpolated from two neighboring input views. We use ground-truth camera poses to calculate the interpolation coefficients, checking whether predicted poses follow the same geometric interpolation rules. We include the details of the interpolation method in Appendix. As shown in Table 5, RayZer demonstrates significantly better performance than PF-LRM and the naive baseline of copying the nearest rendered input view. These results verify that poses predicted by RayZer are interpolatable and 3D-aware.

**Probing the learned camera pose space.** To probe how much actual pose information is learned by RayZer, we follow RUST [63] to fit a lightweight 2-layer MLP head on the pose features. We freeze the camera estimator's transformers and train the MLP under camera supervision. As shown in Table 6, our probing outperforms the supervised baseline (which has the same model architecture and uses

| | Training Supervision | Inference w. GT Pose | Even Sample PSNR$_{\uparrow}$ | Even Sample SSIM$_{\uparrow}$ | Even Sample LPIPS$_{\downarrow}$ | Random Sample PSNR$_{\uparrow}$ | Random Sample SSIM$_{\uparrow}$ | Random Sample LPIPS$_{\downarrow}$ |
|---|---|---|---|---|---|---|---|---|
| **Supervised methods** (inputs are un-posed & use pose annotations during training) | | | | | | | | |
| PF-LRM | 2D + GT Pose | No | 20.63 | 0.819 | 0.160 | 21.27 | 0.827 | 0.154 |
| **Unsupervised methods** (inputs are un-posed & no pose annotations used during training) | | | | | | | | |
| RayZer-copy | 2D | No | 19.56 | 0.812 | 0.159 | 20.17 | 0.820 | 0.150 |
| RayZer | 2D | No | **27.01** | **0.900** | **0.075** | **26.87** | **0.896** | **0.078** |

Table 5. **Evaluating 3D awareness of predicted camera poses** on Objaverse. Unlike Table 3, here we render novel views by interpolating predicted poses of input views, where the interpolation coefficients are calculated from GT poses. This experiment tests whether the learned `SE(3)` poses are geometrically well-defined and 3D-aware. We also compare against a naive baseline "RayZer-copy" that simply copies the nearest rendered input view.

transformers trained from scratch), indicating that RayZer's novel view synthesis self-supervision facilitates a better latent pose space. In contrast, supervised learning struggles due to the challenges of low-dimensional pose representation [5, 11, 44, 90, 97].

### 5.4. Ablation Study

We ablate the main design choices of RayZer from three aspects, including scene representation, 3D prior, and the overall model paradigm. As shown in Table 7 (1), when using the 3DGS representation rather than the latent set representation, the training does not converge. This verifies the optimization difficulty of explicit 3D representation [40, 91] and demonstrates the flexibility of the latent representation with its learned rendering decoder.

Table 7 (2) and (3) ablate the prior of camera representation. Without Plücker ray maps, we observe a degraded performance in (2), showing the effectiveness of using Plücker ray maps to regularize the solution of structure-and-motion problem. Besides, we observe a slightly better performance of (3), which directly uses camera tokens $\mathbf{p}^*$, compared to (2). The reason is that the camera tokens $\mathbf{p}^* \in \mathbb{R}^d$ can leak target image information, while `SE(3)` poses used in (2) serve as an information bottleneck to enforce this disentanglement. Moreover, `SE(3)` poses are geometrically well-defined, allowing us to interpolate them and generate novel views along the interpolated camera trajectory, while the latent camera representation is not directly interpolable.

Table 7 (4) ablates the overall paradigm. When the model first predicts the latent scene and then estimates poses, we observe a degraded performance. In detail, the pose estimator takes the scene representation and target image feature tokens as inputs. The result verifies our insight that pose estimation can be a strong condition for scene reconstruction, championing traditional pose-first methods in the context of self-supervised learning. Note that combining (3) and (4) will be a model that is similar to RUST conceptually.

## 6. Conclusion

We introduce RayZer, a self-supervised large multi-view 3D Vision model trained with zero 3D supervision, *i.e.*, no 3D geometry and camera annotations. RayZer achieves comparable or even better novel view synthesis performance than prior works that use pose labels in both training and inference, verifying the feasibility of breaking free from supervised learning.

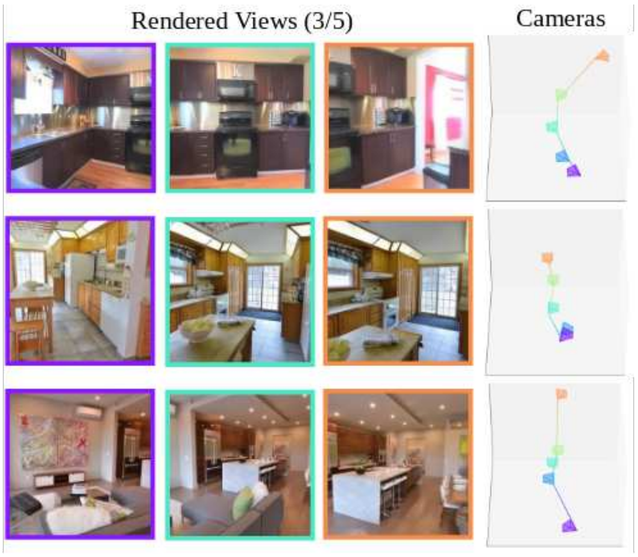

Figure 6. **Visualization of RayZer predicted cameras learned with self-supervision**. We visualize 3 out of 5 rendered views due to space limit, where the image index is highlighted by its color.

| | Pose Encoder ($\mathcal{E}_{pose}$) | *Rotation Acc.*↑ *(%)* R@10° | R@20° | R@30° | *Translation Acc.*↑ *(%)* t@0.1 | t@0.2 | t@0.3 |
|---|---|---|---|---|---|---|---|
| DL3DV | supervised | 39.3 | 63.0 | 77.8 | 15.7 | 33.1 | 44.4 |
| | self-supervised | **47.6** | **72.5** | **84.0** | **20.8** | **44.0** | **60.5** |
| RealEstate | supervised | 87.0 | 96.4 | 99.6 | 44.6 | 59.3 | 82.5 |
| | self-supervised | **99.6** | **99.9** | **100** | **61.2** | **84.2** | **92.8** |
| Objaverse | supervised | 19.8 | 46.7 | 66.0 | 15.1 | 37.2 | 53.8 |
| | self-supervised | **33.6** | **69.2** | **86.8** | **20.1** | **52.7** | **75.5** |

Table 6. **Effectiveness of self-supervised pre-training for pose estimation**. We train a two-layer MLP (with supervised learning) to read out latent camera tokens $\mathbf{p}^*$ predicted by the pose encoder $\mathcal{E}_{pose}$, where the backbone is frozen. At the same time, we also compare with the baseline where both encoder $\mathcal{E}_{pose}$ and the pose prediction MLP are trained with supervised learning from scratch.

| | | Even Sample PSNR | Even Sample SSIM | Even Sample LPIPS | Random Sample PSNR | Random Sample SSIM | Random Sample LPIPS |
|---|---|---|---|---|---|---|---|
| (0) | RayZer | **24.36** | **0.757** | **0.209** | **23.72** | **0.733** | **0.222** |
| (1) | **Representation** - 3DGS + rasterization | – | – | failed | | – | – |
| (2) | **Prior** - no Plücker ray, use `SE(3)` pose | 22.73 | 0.687 | 0.249 | 21.88 | 0.647 | 0.274 |
| (3) | **Prior** - no explicit pose, use latent camera | 23.13 | 0.700 | 0.251 | 22.36 | 0.668 | 0.272 |
| (4) | **Paradigm** - scene first, not pose first | 13.31 | 0.338 | 0.732 | 13.12 | 0.337 | 0.729 |

Table 7. **Ablation study of RayZer designs** on DL3DV with continuous inputs. (1) is a variant uses the 3D Gaussian representation rather than latent scene representation with its learned rendering decoder used by RayZer; (2) does not use Plücker ray maps $\mathcal{R}_{\mathcal{A}}$ for conditioning latent reconstruction. Instead, it encodes the `SE(3)` poses $\mathbf{P}_{\mathcal{A}}$ and intrinsics $\mathbf{K}$ into tokens as condition; (3) directly uses the latent camera tokens $\mathbf{p}^*$, rather than converting it to any explicit forms of cameras, to condition the latent scene reconstruction; (4) first reconstructs latent scene and then estimates pose as Plücker ray maps, contrasting our pose-first paradigm.

## References

[1] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, 1968.

[2] Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan L Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. In *CVPR*, pages 22861–22872, 2024.

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, page 4, 2021.

[4] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. In *CVPR*, pages 4160–4169, 2023.

[5] Romain Brégier. Deep regression on manifolds: a 3d rotation case study. In *2021 International Conference on 3D Vision (3DV)*, pages 166–174. IEEE, 2021.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.

[7] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, pages 5799–5809, 2021.

[8] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, pages 16123–16133, 2022.

[9] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, pages 19457–19467, 2024.

[10] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021.

[11] Jiayi Chen, Yingda Yin, Tolga Birdal, Baoquan Chen, Leonidas J Guibas, and He Wang. Projective manifold gradient layer for deep rotation regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6646–6655, 2022.

[12] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[13] Yuedong Chen, Haofei Xu, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Explicit correspondence matching for generalizable neural radiance fields. *arXiv preprint arXiv:2304.12294*, 2023.

[14] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, pages 370–386. Springer, 2024.

[15] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, pages 628–644. Springer, 2016.

[16] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

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

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[19] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *CVPR*, pages 4970–4980, 2023.

[20] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024.

[21] Yang Fu, Ishan Misra, and Xiaolong Wang. Mononerf: Learning generalizable nerfs from monocular videos without camera poses. In *International Conference on Machine Learning*, pages 10392–10404. PMLR, 2023.

[22] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *CVPR*, pages 20796–20805, 2024.

[23] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, pages 484–499. Springer, 2016.

[24] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2485–2494, 2020.

[25] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *CVPR*, pages 7779–7788, 2020.

[26] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999: 21. DAGM-Symposium Bonn, 15.–17. September 1999*, pages 94–101. Springer, 1999.

[27] Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

[28] 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.

[29] Hanwen Jiang, Zhenyu Jiang, Yue Zhao, and Qixing Huang. Leap: Liberate sparse-view 3d modeling from camera poses. *arXiv preprint arXiv:2310.01410*, 2023.

[30] Hanwen Jiang, Qixing Huang, and Georgios Pavlakos. Real3d: Scaling up large reconstruction models with real-world images. *arXiv preprint arXiv:2406.08479*, 2024.

[31] Hanwen Jiang, Zhenyu Jiang, Kristen Grauman, and Yuke Zhu. Few-view object reconstruction with unknown categories and camera poses. In *2024 International Conference on 3D Vision (3DV)*, pages 31–41. IEEE, 2024.

[32] Hanwen Jiang, Zexiang Xu, Desai Xie, Ziwen Chen, Haian Jin, Fujun Luan, Zhixin Shu, Kai Zhang, Sai Bi, Xin Sun, et al. Megasynth: Scaling up 3d scene reconstruction with synthesized data. *arXiv preprint arXiv:2412.14166*, 2024.

[33] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

[34] Linyi Jin, Richard Tucker, Zhengqi Li, David Fouhey, Noah Snavely, and Aleksander Holynski. Stereo4d: Learning how things move in 3d from internet stereo videos. *arXiv preprint arXiv:2412.09621*, 2024.

[35] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016.

[36] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[37] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, pages 371–386, 2018.

[38] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[39] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *NIPS*, 30, 2017.

[40] 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.

[41] Zihang Lai, Sifei Liu, Alexei A Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. In *ICCV*, pages 9730–9740, 2021.

[42] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022.

[43] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *ECCV*, pages 71–91. Springer, 2024.

[44] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *Advances in Neural Information Processing Systems*, 33:22554–22565, 2020.

[45] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018.

[46] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *ECCV*, pages 178–196. Springer, 2020.

[47] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. *NeurIPS*, 33:11453–11464, 2020.

[48] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, pages 5741–5751, 2021.

[49] 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 *CVPR*, pages 22160–22169, 2024.

[50] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[51] 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.

[52] Siva Karthik Mustikovela, Varun Jampani, Shalini De Mello, Sifei Liu, Umar Iqbal, Carsten Rother, and Jan Kautz. Self-supervised viewpoint learning from image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3971–3981, 2020.

[53] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, pages 7588–7597, 2019.

[54] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[55] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019.

[56] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pages 4195–4205, 2023.

[57] Julius Plucker. Xvii. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, (155): 725–791, 1865.

[58] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, pages 5648–5656, 2016.

[59] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, pages 918–927, 2018.

[60] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

[62] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, pages 6229–6238, 2022.

[63] Mehdi SM Sajjadi, Aravindh Mahendran, Thomas Kipf, Etienne Pot, Daniel Duckworth, Mario Lučić, and Klaus Greff. Rust: Latent neural scene representations from unposed imagery. In *CVPR*, pages 17297–17306, 2023.

[64] Kyle Sargent, Jing Yu Koh, Han Zhang, Huiwen Chang, Charles Herrmann, Pratul Srinivasan, Jiajun Wu, and Deqing Sun. Vq3d: Learning a 3d-aware generative model on imagenet. In *ICCV*, pages 4240–4250, 2023.

[65] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016.

[66] Cameron Smith, David Charatan, Ayush Tewari, and Vincent Sitzmann. Flowmap: High-quality camera poses, intrinsics, and depth via gradient descent. *arXiv preprint arXiv:2404.15259*, 2024.

[67] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68, 1986.

[68] Noah Snavely. Lecture 15: Structure from motion. `https://www.cs.cornell.edu/courses/cs5670/2017sp/lectures/lec15_sfm.pdf`, 2017.

[69] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, pages 776–794. Springer, 2020.

[70] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *CVPR*, pages 4190–4200, 2023.

[71] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, pages 302–310, 2018.

[72] Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *CVPR*, pages 2595–2603, 2019.

[73] Joseph Tung, Gene Chou, Ruojin Cai, Guandao Yang, Kai Zhang, Gordon Wetzstein, Bharath Hariharan, and Noah Snavely. Megascenes: Scene-level view synthesis at scale. In *European Conference on Computer Vision*, pages 197–214. Springer, 2024.

[74] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. *arXiv preprint arXiv:2311.12024*, 2023.

[75] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, pages 4690–4699, 2021.

[76] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025.

[77] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, pages 20697–20709, 2024.

[78] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality meshes. *arXiv preprint arXiv:2404.12385*, 2024.

[79] Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. *Advances in Neural Information Processing Systems*, 35:3502–3516, 2022.

[80] Chao-Yuan Wu, Justin Johnson, Jitendra Malik, Christoph Feichtenhofer, and Georgia Gkioxari. Multiview compressive coding for 3d reconstruction. In *CVPR*, pages 9065–9075, 2023.

[81] Jianfeng Xiang, Jiaolong Yang, Binbin Huang, and Xin Tong. 3d-aware image generation using 2d diffusion models. In *ICCV*, pages 2383–2393, 2023.

[82] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *NIPS*, 29, 2016.

[83] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025.

[84] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, pages 10371–10381, 2024.

[85] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *NeurIPS*, 37:21875–21911, 2025.

[86] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021.

[87] Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 8(5):2890–2897, 2023.

[88] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023.

[89] Jason Y Zhang, Deva Ramanan, and Shubham Tulsiani. Relpose: Predicting probabilistic relative rotation for single objects in the wild. In *ECCV*, pages 592–611. Springer, 2022.

[90] Jason Y Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. *arXiv preprint arXiv:2402.14817*, 2024.

[91] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *arXiv preprint arXiv:2404.19702*, 2024.

[92] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. *arXiv preprint arXiv:2502.12138*, 2025.

[93] Tianyuan Zhang, Zhengfei Kuang, Haian Jin, Zexiang Xu, Sai Bi, Hao Tan, He Zhang, Yiwei Hu, Milos Hasan, William T. Freeman, Kai Zhang, and Fujun Luan. Relitlrm: Generative relightable radiance for large reconstruction models, 2024.

[94] Zhoutong Zhang, Forrester Cole, Zhengqi Li, Michael Rubinstein, Noah Snavely, and William T Freeman. Structure and motion from casual videos. In *ECCV*, pages 20–37. Springer, 2022.

[95] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, pages 1851–1858, 2017.

[96] 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.

[97] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.

[98] 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. *arXiv preprint 2410.12781*, 2024.

## A. Experimental Details

In this section, we introduce more details of RayZer.

**Objaverse Data Details**. We render Objaverse as continuous videos for training and evaluation. The frames are rendered with corresponding cameras on a unit sphere with a constant distance to the object center. Specifically, we render about 70 frames for azimuth 0° to 360°, where the elevation is randomly sampled between -20° to 60° for each shape instance. We sample frames with the distance between the first frame and the last frame being 50 to 65, covering the camera azimuth rotation for about one cycle.

**Camera Interpolation Details**. For the experiment of interpolating predicted cameras, we use Spherical Linear Interpolation (Slerp) for interpolating the camera pose rotation. This is based on the fact that the camera of Objaverse is moving at a constant speed. Thus, Slerp ensures the correct rotation interpolation. We then find the location on the unit sphere that corresponds to this interpolated rotation angle. Thus, we ensure the interpolated cameras are still on the unit sphere, which matches the camera sampling rule for rendering. In conclusion, this interpolation assumes that 1) the camera is moving in a constant speed, and 2) the rule of sampling camera location is known. Thus, this interpolation is only applicable to the synthetic Objaverse data, and does not apply to DL3DV and RealEstate.

**More Training Details**. For all transformer layers in RayZer, we apply QK-Norm [27] to stabilize the training. We use a latent dimension of 768 for RayZer and all baselines methods. RayZer and LVSM both use a latent set scene representation with 3072 tokens. We use mixed precision training [50] with BF16, further accelerated by FlashAttention-V2 [16] of xFormers [42] and gradient checkpointing [12].

We train RayZer and all baselines with the same training protocol. We use 32 A100 GPUs with a total batch size of 256. During training, we warm up with 3000 iterations, using a linearly increased learning rate from 0 to $4e-4$. We apply a cosine learning rate decay, while the final learning rate is $1.5e-4$. We train all baselines with $50,000$ steps. We clip the gradient with norm larger than 1.0. We follow all other hyper-parameters of LVSM.

**More Model Details**. Following LVSM, we do not use bias terms in linear and normalization layers. We also apply the depth-wise initialization for transformer layers.

**Ablation details**. In Table 7 (2), we use a two-layer MLP to encode the camera pose and intrinsics back to a latent pose representation in $\mathbb{R}^d$. In detail, for the predicted pose of each image (in 6D representation [97]), and the camera intrinsics (as the 4-dimensional focal length and principal points of x-axis and y-axis), we first concatenate them, getting a 10-dimensional pose representation. Then, we use the MLP to map it as a high-dimensional pose feature token. To predict the target views, we use a set of learnable patch-aligned spatial tokens shared across all target images as the

| | | Even Sample | | | Random Sample | | |
|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS |
| (0) | RayZer | **24.36** | **0.757** | **0.209** | **23.72** | **0.733** | **0.222** |
| (1) | first frame as canonical | 23.86 | 0.736 | 0.224 | 23.78 | 0.737 | 0.225 |
| (2) | no curriculum | 23.87 | 0.734 | 0.226 | 23.87 | 0.735 | 0.226 |

Table 8. **Ablation study of RayZer techniques to train on continuous video frames**. (1) is a variant choosing the first image in the sequence as the canonical view, rather than choosing the middle frame. (2) does not use the frame sampling curriculum.

initialization. Thus, the rendering decoder takes in the spatial tokens, the scene tokens, and the pose token. After using transformer for updating, we use the updated spatial tokens to regress the pixel values.

## B. RayZer Training with Continuous Inputs

RayZer takes in multi-view image inputs, which can be sampled from either continuous video frames or an unordered image set. In this section, we present two design choices to improve self-supervised learning on video frames input.

**Canonical View Selection**. Prior works [29, 74] usually select the first image in an image sequence as the canonical view. In contrast, we select the frame at the middle time-step as canonical. In this context, the pose prediction MLP$_{pose}$ initialized with a zero mean for its weights will have a small pose data variance. Otherwise, when using the first frame as canonical, the variance can be much larger. Note that this difference in pose variance can be easily handled with ground-truth camera supervision, thus, prior works choose the first image as the canonical view. However, this is more important for unsupervised methods, like RayZer.

**Curriculum**. We gradually increase the training difficulty by sampling video frames with an increasing distance range. With proper initialization of the model for camera pose estimation, it first learns from images with small camera baselines, benefiting the training with larger camera baselines, that follows. In detail, we use a curriculum with a frame sampling range of 48-64, 96-128, and 24-32 at the beginning of training for DL3DV, RealEstate, and Objaverse, respectively. The frame sampling range is linearly increased to 64-96, 128-192, and 48-65 at the end of training for DL3DV, RealEstate, and Objaverse, respectively. The final frame sampling range is also used for the evaluation. The sampling ranges are set based on the difficulty (camera baseline) of each dataset, following prior works [9, 32, 74, 91, 98].

**Experiments**. We include ablations in Table 8, where removing any of the previously discussed techniques leads to a degraded performance. This demonstrates the effectiveness of our designs of selecting canonical view and using frame sampling curriculum during training.

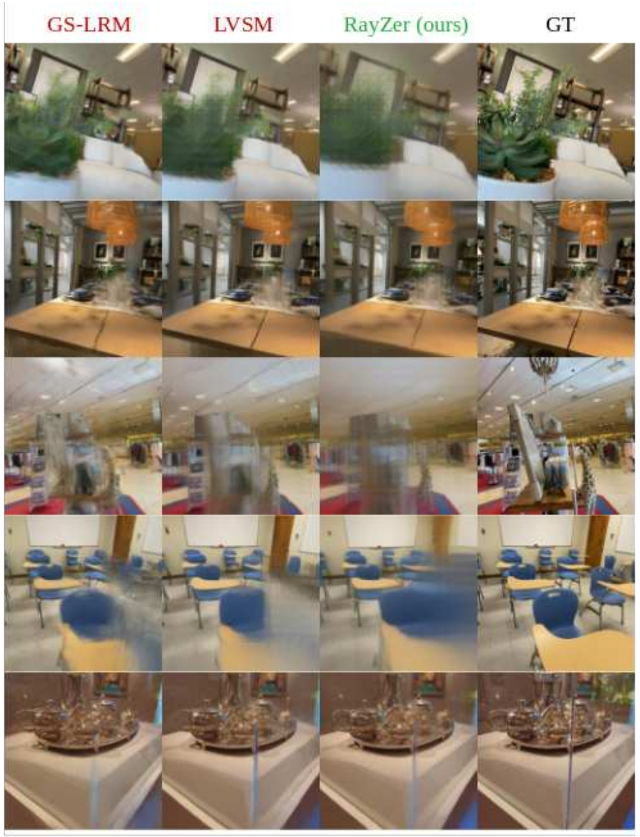

Figure 7. **Visualization of RayZer failure cases** on DL3DV.

## C. More Results

In this section, we present more results for discussing RayZer's failure cases and show more visualizations.

**Failure Case Pattern**. We observe that RayZer can fail when dealing with fine-grained geometry, complicated materials, and occlusions. We present the visualization in Fig. 7. In detail, RayZer fails to handle complicated plant geometry (first row). This failure is not specific to RayZer – GS-LRM and LVSM also can not handle it. In the second and last row, RayZer fails to handle multiple stacked glasses and is not perfect on the specular reflection of the silver teapots. GS-LRM and LVSM also demonstrate imperfect results. In the third and fourth rows, all methods, including RayZer, fail to handle occlusions, where the side view of the exhibition stand is not observed in input views (third row), and the chairs in the fourth row have self-occlusion.

**More Comparisons**. We present more visualization results, comparing with GS-LRM and LVSM in Fig. 8. RayZer generally performs on par, while being a self-supervised method that does not require any camera pose annotations.

**More Visualization**. We present more visualization results comparing with ground-truth novel views in Fig. 9-11.

## D. More Discussion

Why does RayZer demonstrates strong novel view synthesis quality while the fine-tuned pose estimation is not perfect (Table 7 in the main manuscript)? We conjecture RayZer's pose space jointly learns the actual pose information and 3D-aware video frame interpolation at the same time. On datasets with small camera baselines (RealEstate), which is easy to learn, RayZer mainly focuses on learning actual pose estimation. This is supported by the accurate pose estimation performance on RealEstate. On datasets that have large camera baselines (DL3DV and Objaverse), where pose estimation is harder to learn with only self-supervision, RayZer also leverages video interpolation cues together with pose estimation to perform novel view synthesis.

Thus, the method to further enhance disentanglement of interpolation and pose estimation would be an important future direction. In RayZer, using unordered image sets for training and using continuous video frames for training can be two extreme cases in the spectrum for learning this disentanglement. In detail, learning on continuous video frames with using image index positional embeddings strongly encourages the camera pose local smoothness to enhance training performance; while training on unordered image sets fully discards this prior. Finding a balance between the two and designing a better method to encourage the camera pose local smoothness is a promising avenue to solve the structure-and-motion problem with learning `SE(3)` camera poses in the real-world space.

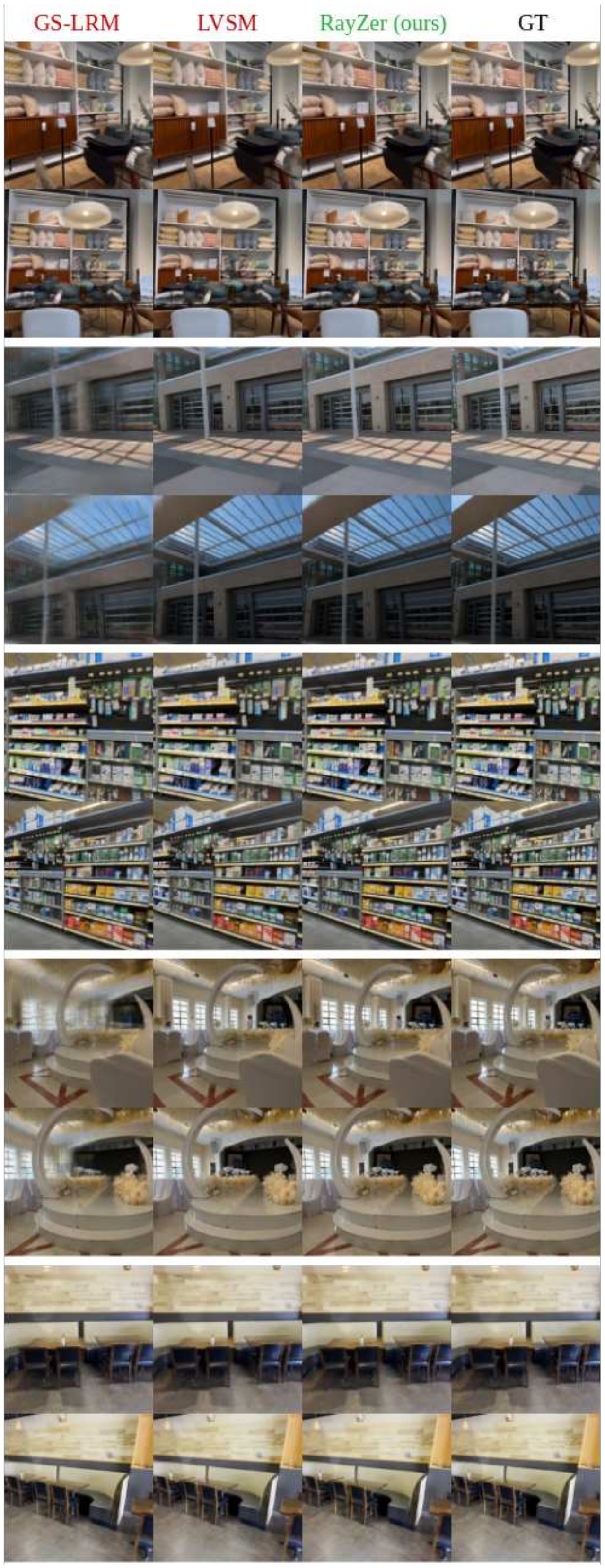

Figure 8. **Visual compression of RayZer and "oracle" methods** on DL3DV.

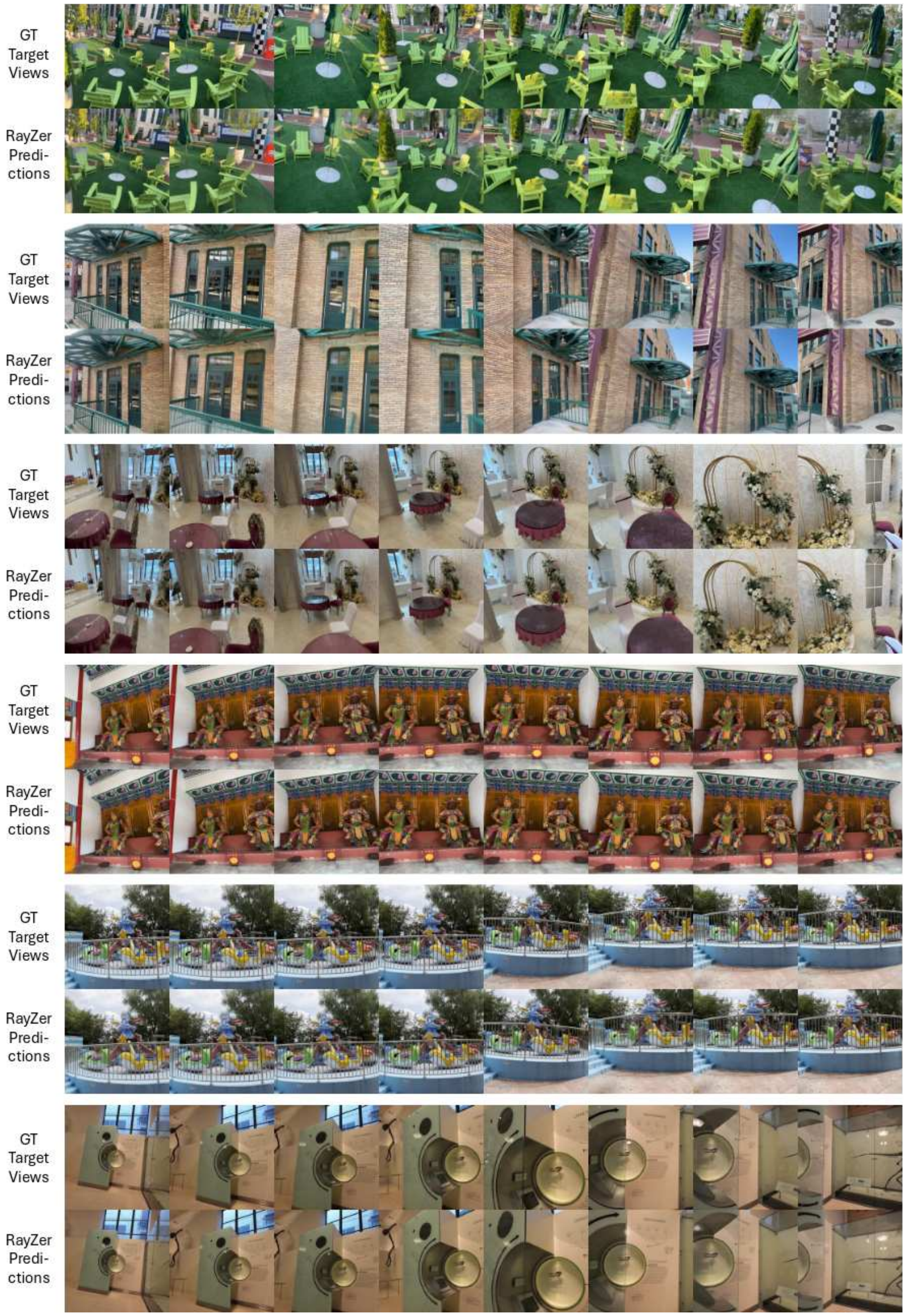

Figure 9. **Visual compression with ground-truth novel views** on DL3DV. The first row of each sample is the target novel views, and the second row are images rendered by RayZer.

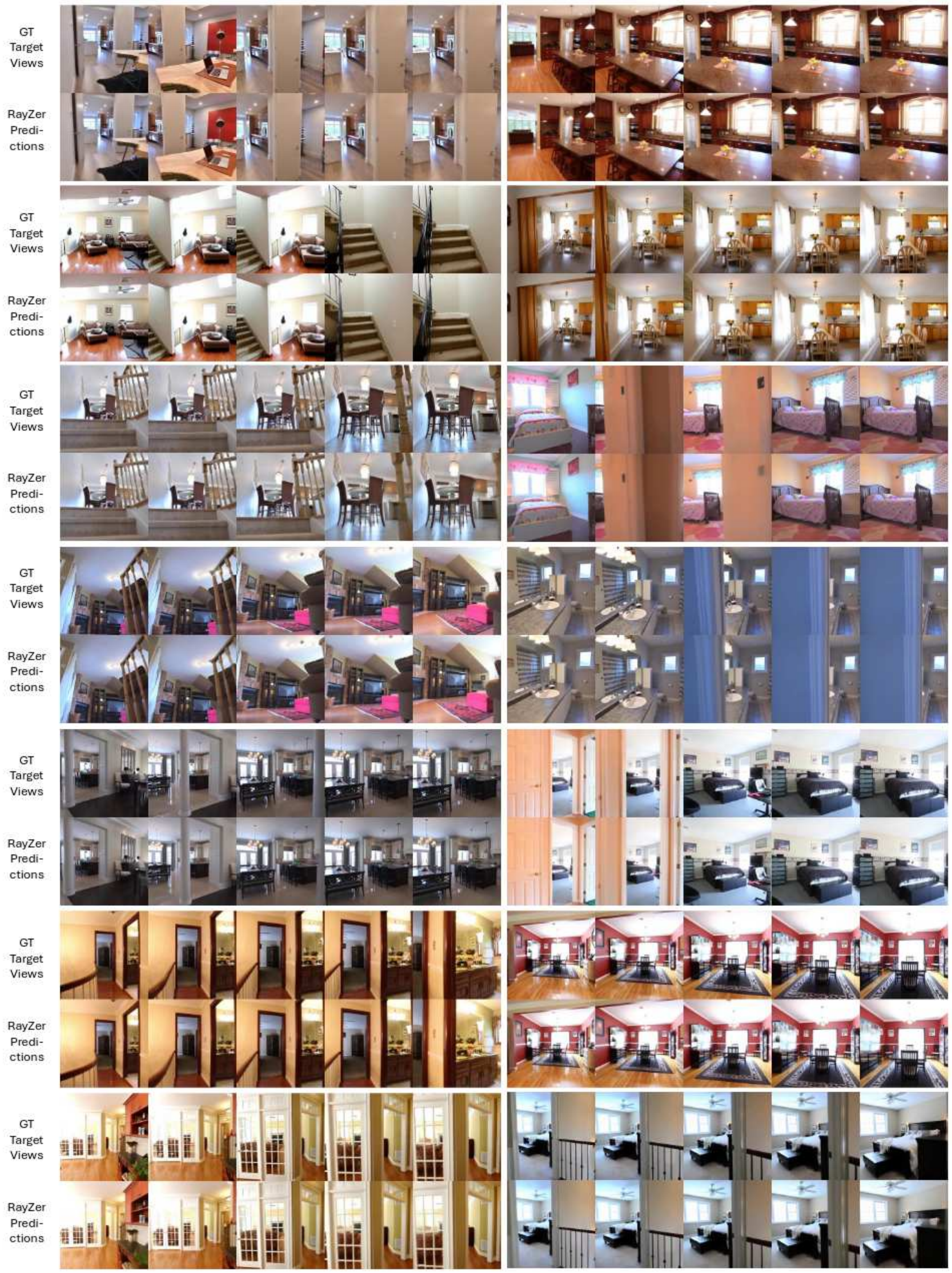

Figure 10. **Visual compression with ground-truth novel views** on RealEstate. The first row of each sample is the target novel views, and the second row are images rendered by RayZer.

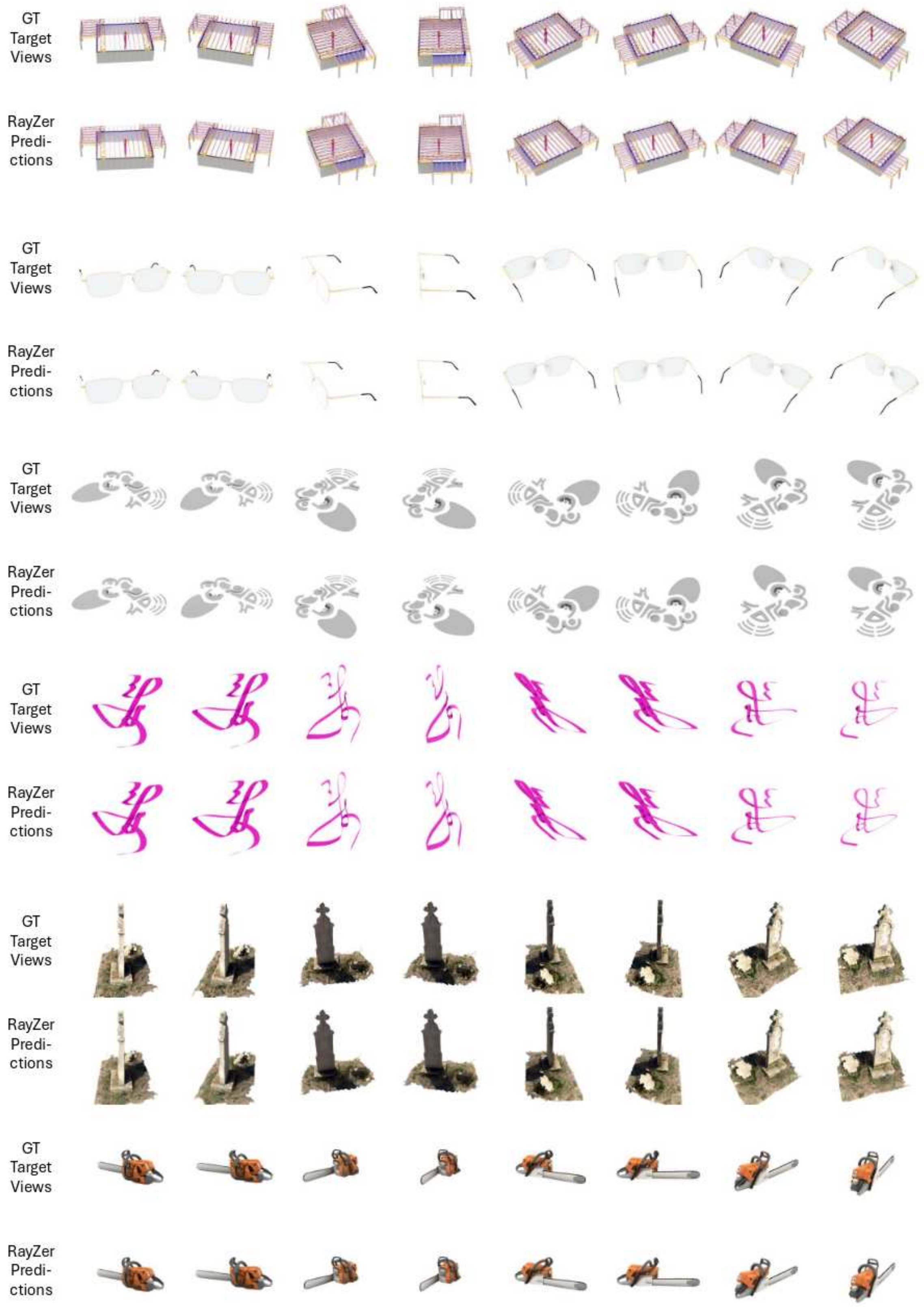

Figure 11. **Visual compression with ground-truth novel views** on Objaverse. The first row of each sample is the target novel views, and the second row are images rendered by RayZer.