ZeroGS: Training 3D Gaussian Splatting from Unposed Images

Yu Chen[1] Rolandos Alexandros Potamias[2] Evangelos Ververas[2]
Jifei Song Jiankang Deng[2] Gim Hee Lee[1]
[1]National University of Singapore [2]Imperial College of London

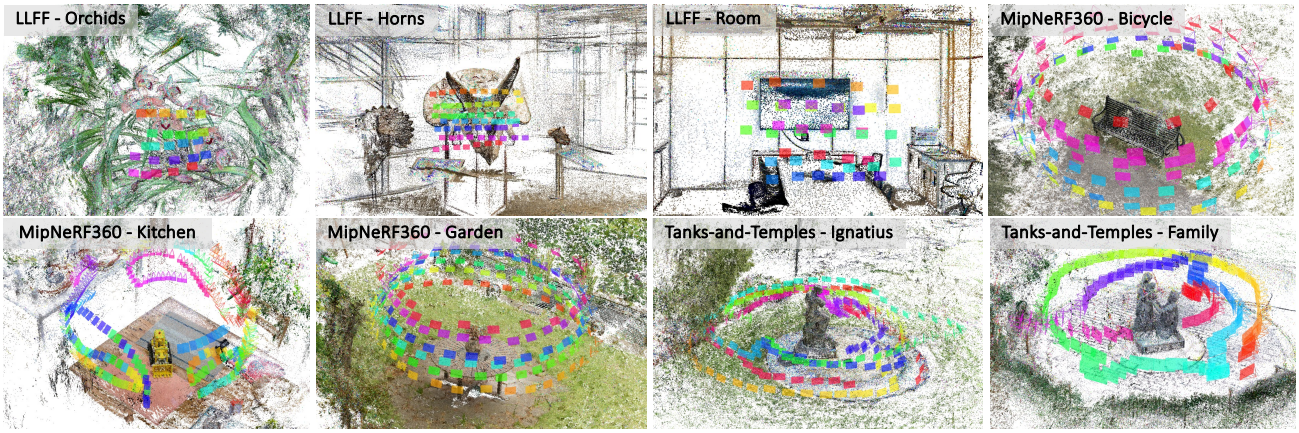

Figure 1. **Reconstruction results of ZeroGS**. Our method reconstructs scenes from hundreds of images without COLMAP poses.

Abstract

Neural radiance fields (NeRF) and 3D Gaussian Splatting (3DGS) are popular techniques to reconstruct and render photo-realistic images. However, the pre-requisite of running Structure-from-Motion (SfM) to get camera poses limits their completeness. While previous methods can reconstruct from a few unposed images, they are not applicable when images are unordered or densely captured. In this work, we propose ZeroGS to train 3DGS from hundreds of unposed and unordered images. Our method leverages a pretrained foundation model as the neural scene representation. Since the accuracy of the predicted pointmaps does not suffice for accurate image registration and high-fidelity image rendering, we propose to mitigate the issue by initializing and finetuning the pretrained model from a seed image. Images are then progressively registered and added to the training buffer, which is further used to train the model. We also propose to refine the camera poses and pointmaps by minimizing a point-to-camera ray consistency loss across multiple views. Experiments on the LLFF dataset, the MipNeRF360 dataset, and the Tanks-and-Temples dataset show that our method recovers more accurate camera poses than state-of-the-art pose-free NeRF/3DGS methods, and even renders higher quality images than 3DGS with COLMAP poses. Our project page is available at aibluefisher.github.io/ZeroGS.

1. Introduction

The renaissance of 3D reconstruction motivates many applications in recording real-world scenes and reconstructing them into a 3D digital world. Most travelers and tourists habitually take pictures and record videos at tourist attractions. These pictures and videos are often uploaded onto websites or applications (such as Niantic Scaniverse[1] and PolyCam[2]) and then reconstructed to 3D models. Behind these applications, neural radiance fields (NeRF) [40] and 3D Gaussian Splatting (3DGS) [28] are increasingly becoming the most popular techniques used to reconstruct 3D scenes. While NeRF/3DGS can reconstruct photo-realistic scenes, they require accurate camera poses from Structure-from-Motion (SfM) tools, *e.g.*, COLMAP [47]. NeRF/3DGS struggles on initialization with inaccurate camera poses and often produces blurry images. Existing methods try to optimize the inaccurate camera poses jointly with per-scene NeRF [15, 16, 20, 33], or train a generalizable NeRF model [12] or few-shot NeRF without ground-truth camera poses [5]. However, few works try to solve the same problem with 3DGS. While CF-3DGS [21] can train 3DGS without relying on camera poses from COLMAP, it only works on short sequential images where camera poses do not change significantly between consecutive frames.

More recently, the 3D-vision foundation model

[1] https://nianticlabs.com/news/scaniverse4
[2] https://poly.cam/

DUSt3R [59] has motivated methods to decouple camera poses from the training process of generalizable 3DGS [49, 64]. DUSt3R takes an image pair as input and outputs pairwise pointmaps in the coordinate frame of the reference image. DUSt3R is trained with massive real-world data containing accurate camera poses and 3D points and generalizes very well to unseen image pairs. However, these generalizable 3DGS methods [49, 64] can only handle the few-shot setting of an image pair since they heavily rely on DUSt3R. InstantSplat [18] leverages the pretrained DUSt3R as a offline tool. Given pairwise images from the same scene, InstantSplat first computes the pairwise pointmaps with DUSt3R, and then aligns the dense pointmaps by jointly optimizing the camera poses and dense points into a global coordinate frame. Subsequently, InstantSplat trains 3DGS with the aligned dense points from DUSt3R. However, the huge GPU memory requirement in optimizing dense pointmaps limits InstantSplat to scenes with few images.

In this work, we propose **ZeroGS** to train 3DGS without relying on COLMAP camera poses. Unlike CF-3DGS and InstantSplat which can only be used for short image sequences or a few images, our method can reconstruct scenes from hundreds of unordered images (*cf*. Fig. 1). Specifically, we leverage a pretrained 3D foundation model as our neural scene representation. In addition to predicting pointmaps, we extend the model to predict the properties of 3D Gaussian primitives. The pretrained model has learned coarse scene geometry priors, making it much easier to jointly optimize the model and camera parameters from scratch. After defining our neural scene representation, we adopt an incremental training pipeline to finetune our model. We first register images by computing coarse camera poses with RANSAC and PnP using the predicted pointmaps in the global coordinate frame. The coarse camera poses are then refined by a point-to-camera ray consistency loss, and our model is further finetuned on the newly registered images. We repeat the process until all images are registered. In this way, our incremental training pipeline is similar to the classical incremental SfM method [47] but differs as follows:

- **Seed Initialization**. Incremental SfM initializes from an image pair, where the camera poses of the image pair are fixed after initialization to fix the gauge freedom. However, our method initializes from a pretrained model and only one seed image.
- **Image Registration**. Instead of registering images individually in an incremental manner, our method registers a batch of images each time. The registered images are further utilized to finetune the neural model.
- **Objective Function**. We finetune our model using a rendering loss but scenes are optimized by the reprojection error in SfM.
- **Scene Sparsity**. Our model predicts dense scene geometries, while SfM outputs sparse scene structures.

Our incremental training pipeline also shares some similarities with a recent learning-based SfM method ACE0 [9], with several key differences: 1) ACE0 only predicts sparse pointmaps, while our method predicts dense pointmaps and 3D Gaussian primitives. 2) ACE0 uses 2D CNN and MLP as the neural scene representation, while we use transformers as the scene representation. 3) The training batch of ACE0 is composed of pixels from multiple views, while our method takes as input image pairs in the training batch. Moreover, finetuning a pretrained foundation model such as DUSt3R is not easy. This is because these 3D foundation models are supervised by ground-truth 3D points that are difficult to obtain in unseen scenes.

We evaluated our method on the LLFF [41] dataset, the MipNeRF360 [3] dataset, and the Tanks-and-Temples dataset [31] and compared them to state-of-the-art pose-free NeRF/3DGS methods. The experimental results show that our method is the best among these methods and our image rendering quality is even better than training NeRF/3DGS from the COLMAP poses.

2. Related Work

Neural Radiance Fields. Neural radiance fields [40] enable rendering from novel viewpoints with encoded frequency features [52]. Many follow-up works try to improve the rendering and training efficiency [11, 19, 35, 66] by encoding scenes into sparse voxels, multi-resolution hash tables [42], or three orthogonal axes and planes. Another branch of NeRF methods focuses on generalizable NeRF [10, 27, 36, 51, 58], and alleviating the aliasing issue by approximating the cone sampling into the scale-aware integrated positional encodings [2] for vanilla NeRF or hexagonal sampling [4] for Instant-NGP [42]. Works are also done in registering multiple blocks of NeRF using traditional optimization method [22] or pretrain a generalizable geometry-aware transformer [13] from 3D data. Despite the limitation of training on small-scale scenes, the divide-and-conquer strategy is adopted to handle city-scale scenes [39, 45, 53, 54, 62].

To remove SfM poses from the training pipeline, NeRF−− [60] jointly optimizing the network of NeRF and camera pose embeddings, SiNeRF [61] adopts a SIREN-MLP [48] and a mixed region sampling strategy to circumvent the sub-optimality issue in NeRF−−. BARF [33] proposes joint training of NeRF with imperfect camera poses from coarse-to-fine, where high-frequencies are progressively activated during training to alleviate the gradient inconsistency issue. GARF [16] extends BARF with the Gaussian activation, enabling training a positional-embedding less coordinate network. RM-NeRF [26] jointly trains a GNN-based motion averaging network [23, 43] and

Mip-NeRF [2] to solve the camera pose refinement issue in multi-scale scenes. However, all the above methods can only handle simple scenes (*e.g.*, forward-facing cameras only) or require accurate pose priors to converge.

3D Gaussian Splatting. Different from NeRF which uses volume rendering to infer the scene occupancy, 3D Gaussian Splatting [28] (3DGS) initializes 3D Gaussians from a sparse point cloud and renders scenes by differentiable rasterization, and can achieve real-time rendering performance. However, 3DGS can face difficulty in identifying more Gaussians when initialized from textureless areas. To encourage the learning of a better scene geometry, Scaffold-GS [38] initializes a sparse voxel grid from the initial point cloud and encodes the features of 3D Gaussians into corresponding feature vectors. The introduction of the sparse voxel reduces the Gaussian densities by avoiding unnecessary densification on the scene surface. SAGS [56] implicitly encodes the scene structure into a GNN. Other works also try to learn 2D Gaussians [25] to fit the scene surface [24] more accurately. Similar to NeRF, 3DGS faces the aliasing issue caused by the fixed window Gaussian kernel during rasterization. The same issue is handled in Mip-Splatting [67] and many latter works [32, 50, 63]. Vast-Gaussian [34] and its follow-ups [14, 29, 37] focus on developing distributed training methods to reconstruct the large-scale scenes.

Although 3DGS can render higher-fidelity images, it relies on accurate camera poses. Unlike NeRF where many works have been proposed to solve the inaccuracy of camera poses, the same issue has not been widely tackled in 3DGS. The most relevant work to ours is CF-3DGS [21] and InstantSplat [18]. However, CF-3DGS requires depth estimation to initialize the 3D Gaussians, and it can only optimize camera poses and 3DGS between short consecutive image sequences. CF-3DGS is highly susceptible to failure when camera poses change significantly or images are unordered. InstantSplat [18] leverages an existing pretrained 3D foundation model DUSt3R [59] to regress dense pointmaps between image pairs, followed by obtaining the camera poses by aligning the pointmaps into a global coordinate frame. However, aligning dense pointmaps is time- and memory-consuming. As a result, InstantSplat can only handle a very few images. Other related work includes Splatt3R [49] which extends DUSt3R to predict 3D Gaussians without posed images. Nonetheless, it works only on image pairs since DUSt3R produces pointmaps in the reference frame of the first image instead of a global coordinate frame. Our work also leverages a pretrained DUSt3R-based network, *i.e.* Spann3R [57]. However, unlike InstantSplat and Splatt3R, our method regresses points in the global coordinate frame of a seed image without aligning the dense pointmaps, followed by incrementally registering images and finetuning the network. Our method can handle hundreds of images and run on a 24GB consumer-level GPU.

3. Method

In this section, we first give the preliminaries of our scene regressor network followed by introducing our training pipeline (see Fig. 2) for incrementally reconstructing a scene. We first use Spann3R as the scene regressor network to predict 3D Gaussians $\mathbf{G}_k$ and pointmaps $\mathbf{X}_k$ from an image pair. We then use RANSAC and a PnP solver to obtain the initial camera poses based on 2D-3D correspondences. Furthermore, we refine the coarse camera poses by minimizing a point-to-camera ray consistency loss between 3D points and camera centers. Subsequently, we rasterize the 3D Gaussians with the refined camera poses to render images. An RGB loss is adopted for back-propagating gradients. At the end of each training epoch, we update our training buffer by registering new images with RANSAC and a PnP solver. This process is repeated until all images are registered or no more images can be registered.

3.1. Preliminaries

Our network architecture is based on DUSt3R [59] and Spann3R [57]. Given an image pair $(\mathbf{I}_i, \mathbf{I}_j)$, DUSt3R predicts the corresponding pointmaps $(\mathbf{X}^{i,i}, \mathbf{X}^{j,i})$ for each image, where $\mathbf{X}^{i,j}$ denotes the pointmap $\mathbf{X}_j$ expressed in camera i's coordinate frame.

DUSt3R uses a ViT [17] as a shared encoder for both images and two transformer decoders for the reference image i and the target image j, respectively. The two decoders denoted as *reference decoder* $\mathcal{D}_{\text{ref}}$ and *target decoder* $\mathcal{D}_{\text{tgt}}$ consist of two projection heads $\mathcal{H}_{\text{ref}}, \mathcal{H}_{\text{tgt}}$ that map the decoder features into pointmaps:

$$\begin{aligned} &\mathbf{f}_i^{\text{e}}, \mathbf{f}_j^{\text{e}} = \mathcal{V}(\mathbf{I}_i, \mathbf{I}_j), \mathbf{f}_i^{\text{d}} = \mathcal{D}_{\text{ref}}(\mathbf{f}_i^{\text{e}}, \mathbf{f}_j^{\text{e}}), \mathbf{f}_j^{\text{d}} = \mathcal{D}_{\text{tgt}}(\mathbf{f}_j^{\text{e}}, \mathbf{f}_i^{\text{e}}), \quad (1) \\ &\mathbf{X}^{i,i},\ \mathbf{X}^{j,i} = \mathcal{H}_{\text{ref}}(\mathbf{f}_i^{\text{d}}),\ \mathcal{H}_{\text{tgt}}(\mathbf{f}_j^{\text{d}}). \end{aligned}$$

DUSt3R reconstructs image pairs in a local coordinate frame. When handling more than two images, DUSt3R uses a post-processing step to align the pairwise dense pointmaps to a global coordinate frame, which is time-consuming and can exceed the GPU memory limitation.

Spann3R proposes a feature fusion mechanism to predict pointmaps $(\mathbf{X}^{i,g},\ \mathbf{X}^{j,g})$ in a global coordinate frame. It computes a fused feature $\mathbf{f}_t^G$ in the t-th training epoch from a spatial feature memory. The reference decoder inputs the fused feature for reconstruction and the target decoder produces features for querying the memory. Furthermore, Spann3R uses two additional projection heads to compute the key and query feature for reconstructing the next image pairs, and a memory encoder $\mathcal{V}_{\text{mem}}$ which encodes the

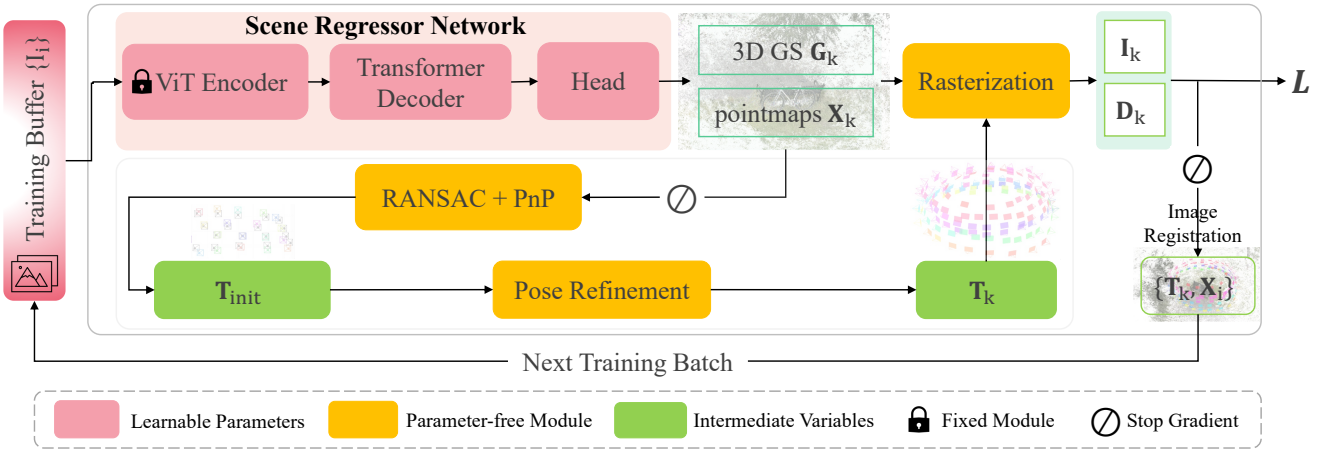

Figure 2. **The training pipeline of our Pose-Free 3D Gaussian Splatting**. Our method follows the classical incremental SfM reconstruction pipeline with the key difference that the input is no longer an image but a pair of images in a progressively updated training buffer. The scene regressor network is trained as follows: 1) Use Spann3R [57] as the scene regressor network to predict 3D Gaussians $\mathbf{G}_k$ and pointmaps $\mathbf{X}_k$ from a pair of images. 2) Leverage RANSAC and a PnP solver to obtain the initial camera poses based on direct 2D-3D correspondences. 3) Refine the coarse camera poses by minimizing the point-to-ray consistency loss between 3D tracks and camera centers. 4) Rasterize the 3D Gaussians with the refined camera poses to render images. An RGB loss is adopted for back-propagating gradients. 5) After each training epoch, we update the training buffer by registering more images.

pointmaps from the reference decoder:

$$\mathbf{f}_j^Q = \mathcal{H}_{\text{tgt}}^{\text{Q}}(\mathbf{f}_j^d),\ \mathbf{f}_i^K = \mathcal{H}_{\text{ref}}^{\text{K}}(\mathbf{f}_i^{\text{d}}),\ \mathbf{f}_i^V = \mathcal{V}_{\text{mem}}(\mathbf{X}^{i,g}). \quad (2)$$

Although Spann3R can reconstruct out-of-distributed scenes, it reconstructs images individually and is limited to very short frames due to GPU memory limitation. Moreover, the 3D points predicted by Spann3R in these scenes lack accuracy. We refer readers to [57, 59] for more details.

3.2. Neural Scene Representation

In 3DGS, scenes are explicitly represented by a set of 3D Gaussian primitives [28] or implicitly represented by neural networks [38, 46]. In this work, we use Spann3R [57] as neural scene representation and extend it to also predict Gaussian primitives. We refer to our neural scene representation as the *scene regressor network* f_{SCR}, which analogs to the scene regressor in pose regression or localization networks [8, 9]. Unlike the sparse scene regressor that takes an image as input, our scene regressor takes an image pair as input and predicts dense pointmaps $\mathbf{X}_i$ and per-pixel 3D Gaussians $\mathbf{G}_i$ in a global coordinate frame:

$$(\mathbf{X}_i, \mathbf{G}_i;\ \mathbf{X}_j, \mathbf{G}_j) = f_{\text{SCR}}(\mathbf{I}_i, \mathbf{I}_j), \quad (3)$$

where $\mathbf{I}_i$ is the reference and $\mathbf{I}_j$ is the target image.

By rasterizing the set of 3D Gaussian primitives $\mathcal{G} = \{\mathbf{G}_i\}$, we back-propagate gradients to the model using an RGB loss. More specifically, a 3D Gaussian primitive is composed of the opacity o, the mean $\mathbf{u}$, the covariance $\mathbf{\Sigma}$, and the coefficients of the spherical harmonics $\mathbf{SH}$. The covariance is decomposed into a rotation matrix $\mathbf{R}$ and a scaling matrix $\mathbf{S}$ to ensure the positive semi-definiteness: $\mathbf{\Sigma}_i = \mathbf{RSS}^\top\mathbf{R}^\top$. In addition, instead directly predicting the mean $\mathbf{u}$ for each 3D Gaussian, we predict an offset $\Delta\mathbf{X}$ and apply it to the pointmaps to obtain the mean $\mathbf{u} = \mathbf{X} + \Delta\mathbf{X}$. To render the color for a pixel $\mathbf{p}$, the 3D Gaussians are projected into the image space for alpha blending:

$$\mathbf{C} = \sum_i \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \quad (4)$$

where α_i is the rendering opacity and is computed by $\alpha = o \cdot \mathbf{G}^{\text{proj}}(\mathbf{p})$, $\mathbf{c}_i$ is the per-pixel color that computed from the spherical harmonics $\mathbf{SH}$. In practice, Eq. (4) is computed using a differentiable rasterizer [28]:

$$\hat{\mathbf{I}} = \mathcal{R}(\mathbf{T}, \mathbf{K}; o, \mathbf{R}, \mathbf{S}, \mathbf{SH}) = \mathcal{R}(\mathbf{T}, \mathbf{K}; \{\mathbf{G}_i\}), \quad (5)$$

where $\mathbf{T}$ is the camera extrinsics and $\mathbf{K}$ is the intrinsics.

3.3. Incremental Reconstruction

We incrementally reconstruct each scene with the scene regressor as the neural representation. We emphasize that finetuning a pre-trained model such as DUSt3R or Spann3R on unseen scenes is challenging. This is because the existing DUSt3R-based model is trained with ground-truth 3D points. However, obtaining ground-truth 3D points can be expensive and impossible in most scenes.

3.3.1. Seed Initialization

Given a set of unordered images $\mathcal{I} = \{\mathbf{I}_i\}$, we first select a *seed image* for initialization. This is different from incremental SfM, which requires a *seed image pair* for two-view

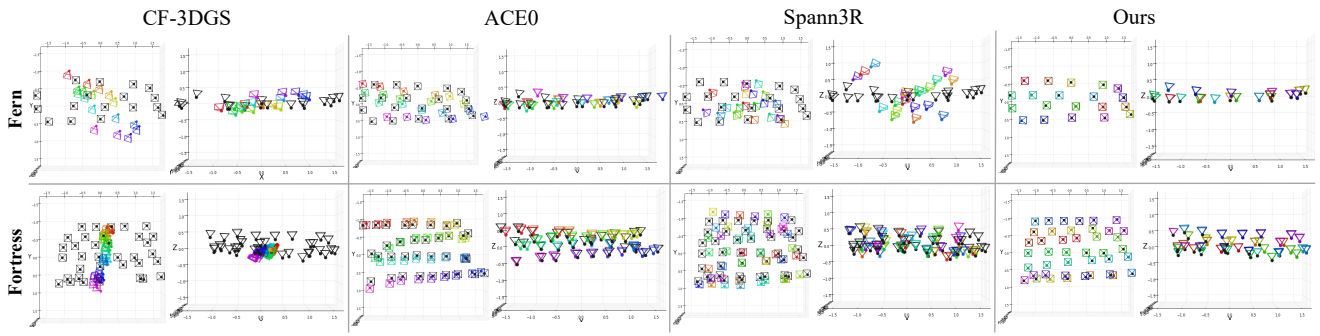

Figure 3. **Visualization of camera poses accuracy** on the LLFF dataset (Zoom in for best view). Black: pseudo-ground-truth camera poses obtained from COLMAP [47]. Colored: predicted camera poses.

reconstruction. The seed image pair must have sufficient matching inliers and a wide baseline. This criterion guarantees the initial pair overlaps with as many other images as possible for later registration. To achieve a similar goal, we use NetVLAD [1] to compute a global descriptor for each image, and then we compute the similarity score between each image pair. We further build a similarity graph $\mathcal{G}_{\text{sim}}$, where the node represents an image, the edge represents the image pair, and the edge weight represents the similarity score. An edge is discarded if its weight is less than a threshold s_{sim}. We then select the node that has the maximum degree as the seed image $\mathbf{I}_{\text{seed}}$. Intuitively, a node with a maximum degree means it has the most adjacent images, which is beneficial for the batched registration of images.

After selecting the seed image, we finetune the scene regressor in a self-supervised manner. Specifically, we set the seed image as the reference frame, and the camera pose of the seed image, $\mathbf{T}_{\text{seed}}$, is set to an identity matrix. We then compute a RGB loss:

$$\mathcal{L}_{\text{rgb}} = \sum \|\mathbf{I} - \hat{\mathbf{I}}\|_1 = \sum \|\mathbf{I} - \mathcal{R}(\mathbf{T}_{\text{seed}}, \mathbf{K}, \{\mathbf{G}_i\})\|_1. \quad (6)$$

Note that, during initialization, the seed image serves as both the reference and target image to the scene regressor: $(\mathbf{X}_i, \mathbf{G}_i;\ \mathbf{X}_j, \mathbf{G}_j) = f_{\text{SCR}}(\mathbf{I}_{\text{seed}}, \mathbf{I}_{\text{seed}})$, and the camera pose is fixed as an identity matrix.

3.3.2. Image Registration

After seed initialization, we incrementally register a batch of images $\mathcal{I}_{\text{buf}} = \{\mathbf{I}_k\}$ in a training epoch. We add a batch of newly registered images into the training buffer and train the scene regressor. Upon training convergence, we expand the training buffer by selecting a new batch of images. This process is repeated until all images are registered.

Coarse Camera Pose Estimation. Given a registered reference image $\mathbf{I}_i$ and a to-be-registered target image $\mathbf{I}_k$, we pass them to the scene regressor and obtain the 3D points $\{\mathbf{X}_k\}$ in a global coordinate frame. Since we have the coordinates $\{\mathbf{u}_k\}$ of each image pixel and their corresponding 3D coordinates $\{\mathbf{X}_k\}$, we can easily find the 2D-3D correspondences $\{(\mathbf{u}_k, \mathbf{X}_k)\}$. We then use RANSAC and a PnP solver to obtain a coarse camera pose:

$$\mathbf{T}_k^{\text{coarse}}, S_k = \text{PnP}(\mathbf{K}, \{(\mathbf{u}_k, \mathbf{X}_k)\}), \quad (7)$$

where S_k is the number of inliers and $\mathbf{X}_k = f_{\text{SCR}}(\mathbf{I}_{\text{ref}}, \mathbf{I}_k)$. $\mathbf{I}_{\text{ref}}$ is the reference image and $\mathbf{I}_k$ is the target image we want to register. We add the target image $\mathbf{I}_k$ into the training buffer only when the inlier number is larger than the inlier threshold s_{inlier}. After initialization, the seed image is selected as the reference image. In the following training batches, we select the reference image from the registered images which connects to most of the unregistered images.

Camera Pose Refinement. The camera poses of newly registered images can be inaccurate since the scene regressor has not seen these images. While ACE0 [9] uses a MLP pose refiner to alleviate this issue during training, we experimentally found that does not improve the pose accuracy with our transformer-based scene regressor. This is because ACE0 uses MLP as the scene coordinate decoder, and each pixel is individually mapped onto the 3D space. ACE0 thus enables network training by mixing millions of pixels from different views in a training batch, and the multiple-view constraint helps constrain the network training. However, since we use a transformer-based decoder and due to the GPU memory limitation, we can use only limited views (we use a batch size 1 in practice on a 24GB consumer-level GPU) in each training batch, which can easily lead to the divergence of network training.

To solve the aforementioned issue, we propose to further refine the coarse camera poses by minimizing a point-to-camera ray consistency loss below:

$$\arg\min_{\mathbf{X}_i, \mathbf{C}_k} \sum_{i,k} \rho(\|d_{i,k} \cdot \boldsymbol{\nu}_{i,k} - (\mathbf{X}_i - \mathbf{C}_k)\|_2), \quad (8)$$

where $\mathbf{C}_k$ is the camera center for image $\mathbf{I}_k$, $d_{i,k}$ is the scaling factor between a 3D point $\mathbf{X}_i$ and the camera center $\mathbf{C}_k$, $\boldsymbol{\nu}_{i,k}$ is the ray direction between $\mathbf{X}_i$ and $\mathbf{C}_k$. During optimization, we fix the camera pose of the seed image for the gauge ambiguity and fix the scaling factor between the seed image and its most similar adjacent image for the scale

Scenes	BARF [33]		DBARF [12]		ACE0 [9]		CF-3DGS [21]		Spann3R [57]		Ours	
	ΔR	Δt	ΔR	Δt	ΔR	Δt	ΔR	Δt	ΔR	Δt	ΔR	Δt
Fern	0.19	0.192	0.89	0.341	11.87	0.284	2.81	9.254	39.03	0.767	0.26	0.005
Flower	0.25	0.224	1.39	0.318	10.32	0.103	0.24	2.586	11.91	0.285	0.52	0.011
Fortress	0.48	0.364	0.59	0.229	75.07	0.603	1.28	8.592	08.31	0.152	0.04	0.002
Horns	0.30	0.222	0.82	0.292	07.61	0.233	1.15	2.371	06.98	0.349	0.03	0.001
Leaves	1.27	0.249	4.63	0.855	10.87	0.136	0.33	7.350	44.09	0.801	0.22	0.006
Orchids	0.63	0.404	1.16	0.573	06.24	0.168	1.45	2.772	09.77	0.256	0.24	0.006
Room	0.32	0.270	0.53	0.360	13.19	0.424	1.36	3.336	07.48	0.513	0.03	0.001
Trex	0.14	0.720	1.06	0.463	11.93	0.373	1.56	4.431	32.39	0.758	0.03	0.010

Table 1. **Quantitative results of camera pose accuracy on LLFF dataset**. The red, orange and yellow colors respectively denote the best, the second best, and the third best results. The unit for rotation error is degree.

ambiguity. Moreover, for a new training epoch, we fix the camera poses registered in the previous epoch and only optimize the camera poses registered in the current training epoch to improve the optimization efficiency.

3.3.3. Finalizing Neural Scene Reconstruction

We propose a two-stage strategy to improve the final reconstruction quality when all images have been registered or no more images can be added to the training buffer. The first stage is to optimize all camera poses using Eq. (8). This is because we incrementally register images and errors accumulate during training. In this stage, we only fix the camera pose of the seed image, and camera poses obtained from all previous training epochs are used as initial values for optimization. Since the initial values are accurate enough, the final optimization converges very fast. To further improve the image rendering quality, we proposed to refine the scene details using explicit 3D Gaussian primitives [28] in a second stage. This is because we only used fixed low-resolution images during the training of our scene regressor due to GPU memory limitation. The scene regressor can therefore only represent the coarse scene geometry. In the second stage, we use the same strategy as in [28] for 3D Gaussian densification and pruning during refinement.

4. Experiments

Evaluation Datasets. We evaluate our method on the LLFF dataset [41], the Mip-NeRF360 dataset [3], and the Tanks-and-Temples dataset [31]. The LLFF dataset contains 8 scenes with cameras facing forward, each containing about 20-62 images. The Mip-NeRF360 dataset contains different scenes where cameras are distributed evenly in 360 degrees in the 3D space, each containing about 100-300 images. The Tanks-and-Temples dataset is similar to the Mip-NeRF360 dataset in camera poses and scene scales but with more illumination and appearance changes.

Implementation Details. We initialize the scene regressor network using the pre-trained Spann3R model [57]. During training, we use the image resolution of 512×512 for all the datasets. We use a learning rate of $1e-5$ to finetune the scene regressor. We use $s_{\text{sim}} = 0.3$ to reject edges when building the similarity graph. We use DSAC [6, 7] to compute the camera poses for candidate image registration. Since DSAC supports only a single focal length for both the image x-axis and y-axis, we modify it to use different focal lengths for the image x-axis and y-axis. We set the threshold of inlier number s_{inlier} to $5,000$ and the threshold of reprojection error to be within 6 pixels in DSAC. For the seed initialization, we finetune the scene regressor by 500 iterations. During the incremental training, we finetune the scene regressor by 1,000 iterations on the LLFF dataset and 1,500 iterations on the Mip-NeRF360 dataset. For the novel view synthesis task, images are downsampled by 4 during training and inference.

Results. We first present results on the **LLFF Dataset**. We compare our method with BARF [33] and DBARF [12], which are NeRF-based pose-free methods. We also compare our method to the 3DGS-based pose-free method CF-3DGS [21]. Since InstantSplat [18] always reports an out-of-memory issue on the dataset, we do not compare to it. Besides pose-free NeRF/3DGS-based methods, we also compare with a scene regressor ACE0 [9]. Note that Spann3R and ACE0 do not support novel view synthesis during this paper, and thus we only compare with it in terms of the camera pose accuracy. The quantitative results for camera pose accuracy are presented in Table 1. The unit for rotation error is degree, and the unit for translation error is dimensionless due to the loss of absolute scale in the ground-truth camera poses. The results show that the camera pose accuracy of our method consistently outperforms all other methods. BARF and DBARF are the second-best and third-best methods, suggesting that the pose-free methods in 3DGS are under-explored compared to the pose-free NeRF. While CF-3DGS claims to be COLMAP-free, it behaves badly on this dataset since the camera does not move on a smooth curve. We also present the novel view synthesis results in Table 2. Our method also renders the highest

Scenes	NeRF [40]			BARF [33]			3DGS			CF-3DGS [21]			Ours		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Fern	23.72	0.733	0.262	23.79	0.710	0.311	23.63	0.794	0.136	17.35	0.494	0.428	23.77	0.797	0.114
Flower	23.24	0.668	0.244	23.37	0.698	0.211	26.91	0.829	0.096	20.17	0.622	0.362	25.81	0.812	0.108
Fortress	25.97	0.786	0.185	29.08	0.823	0.132	29.93	0.880	0.078	14.73	0.395	0.460	29.31	0.868	0.084
Horns	20.35	0.624	0.421	22.78	0.727	0.298	26.02	0.862	0.121	15.60	0.412	0.514	26.67	0.882	0.093
Leaves	15.33	0.306	0.526	18.78	0.537	0.353	17.91	0.593	0.205	15.38	0.416	0.398	16.45	0.526	0.270
Orchids	17.34	0.518	0.307	19.45	0.574	0.291	18.98	0.612	0.159	13.80	0.258	0.516	19.55	0.640	0.147
Room	32.42	0.948	0.080	31.95	0.940	0.099	28.96	0.927	0.115	18.36	0.713	0.382	32.37	0.948	0.073
Trex	22.12	0.739	0.244	22.55	0.767	0.206	24.74	0.881	0.145	16.76	0.522	0.434	26.11	0.905	0.084

Table 2. **Quantitative results of novel view synthesis on LLFF dataset**. ↑: higher is better, ↓: lower is better.

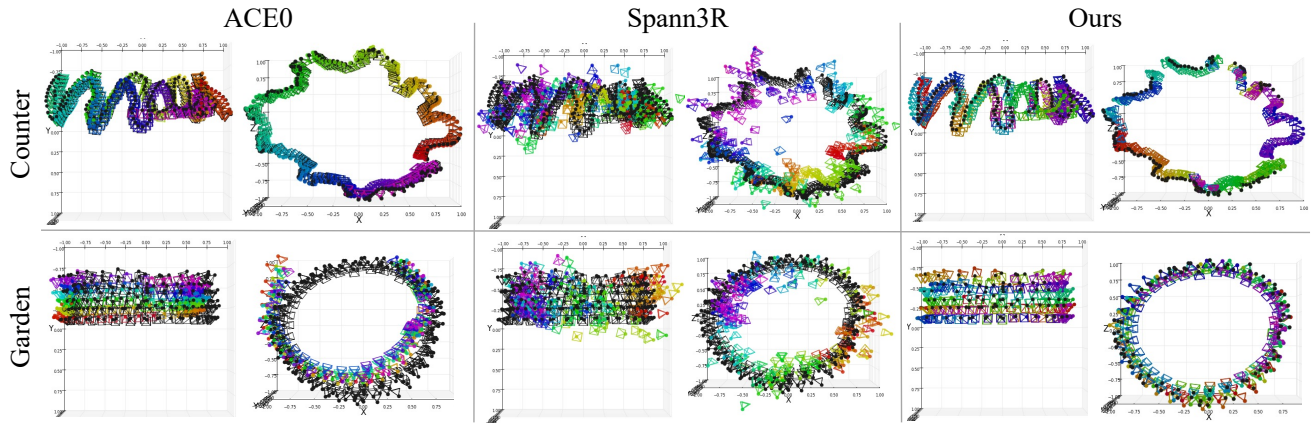

Figure 4. **Visualization of camera poses accuracy** on the MipNeRF360 dataset (Zoom in for best view). Black: pseudo-ground-truth camera poses obtained from COLMAP [47]. Colored: predicted camera poses.

quality images than all other pose-free NeRF/3DGS methods. Surprisingly, our method even surpassed 3DGS trained with COLMAP pose on most scenes. This result suggests our camera pose can be even more accurate than COLMAP in those scenes. We also present the qualitative comparison of the camera pose accuracy in Fig. 3. The ground-truth camera poses are rendered in black, and the predicted camera poses are rendered in rainbow colors. The qualitative results of novel view synthesis are given in Fig. 5.

We also evaluate our method on the **Mip-NeRF360 dataset**. The Mip-NeRF360 dataset is more challenging than the LLFF dataset for pose-free NeRF/3DGS methods, where the latter contains only forward-facing views while the former is composed of cameras that rotate 360 degrees around the complex objects in the scene. Since BARF [33] requires accurate initialization on non-forward-facing datasets and DBARF [12] is designed mainly for scenes that are highly overlapped and have narrow baselines, we do not compare with these methods. We first present the camera pose accuracy in Table 3. We can observe that our method consistently outperforms ACE0 and Spann3R, while CF-3DGS always got an out-of-memory on the dataset, we mark its results by '-'. Some qualitative results of camera poses are shown in Fig. 4.

We further evaluate our method on the **Tanks-and-Temples dataset**. We present the final novel view synthesis results in Table 5. The higher image rendering quality of our method in Table 5 suggests that our camera poses can be more accurate than COLMAP. See our supplementary for more qualitative and quantitative results on this dataset.

Scenes	ACE0 [9]		CF-3DGS [21]		Spann3R [57]		Ours	
	ΔR	Δt	ΔR	Δt	ΔR	Δt	ΔR	Δt
Bicycle	4.56	0.052	-	-	10.79	0.212	0.035	0.005
Counter	1.76	0.017	-	-	11.16	0.226	0.029	0.002
Garden	22.66	0.286	-	-	15.65	0.279	0.028	0.002
Kitchen	2.65	0.044	-	-	9.60	0.171	0.052	0.008

Table 3. **Quantitative results of camera pose accuracy on MipNeRF360 dataset**. The red, orange and yellow colors respectively denote the best, the second best, and the third best results.

Scenes	COLMAP+3DGS			Ours		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Bicycle	22.92	0.695	0.201	23.10	0.707	0.201
Counter	27.64	0.878	0.122	26.87	0.873	0.124
Garden	24.83	0.769	0.153	25.47	0.839	0.107
Kitchen	30.93	0.932	0.054	29.67	0.925	0.061

Table 4. **Quantitative results of novel view synthesis on MipNeRF360 dataset**. ↑: higher is better, ↓: lower is better.

Ablation Study. We ablate the effectiveness of our incremental training step and camera pose refinement step in Table 6. We can see that with the refinement step, the camera pose accuracy is improved significantly. Although our proposed method is initialized from the pre-trained Spann3R

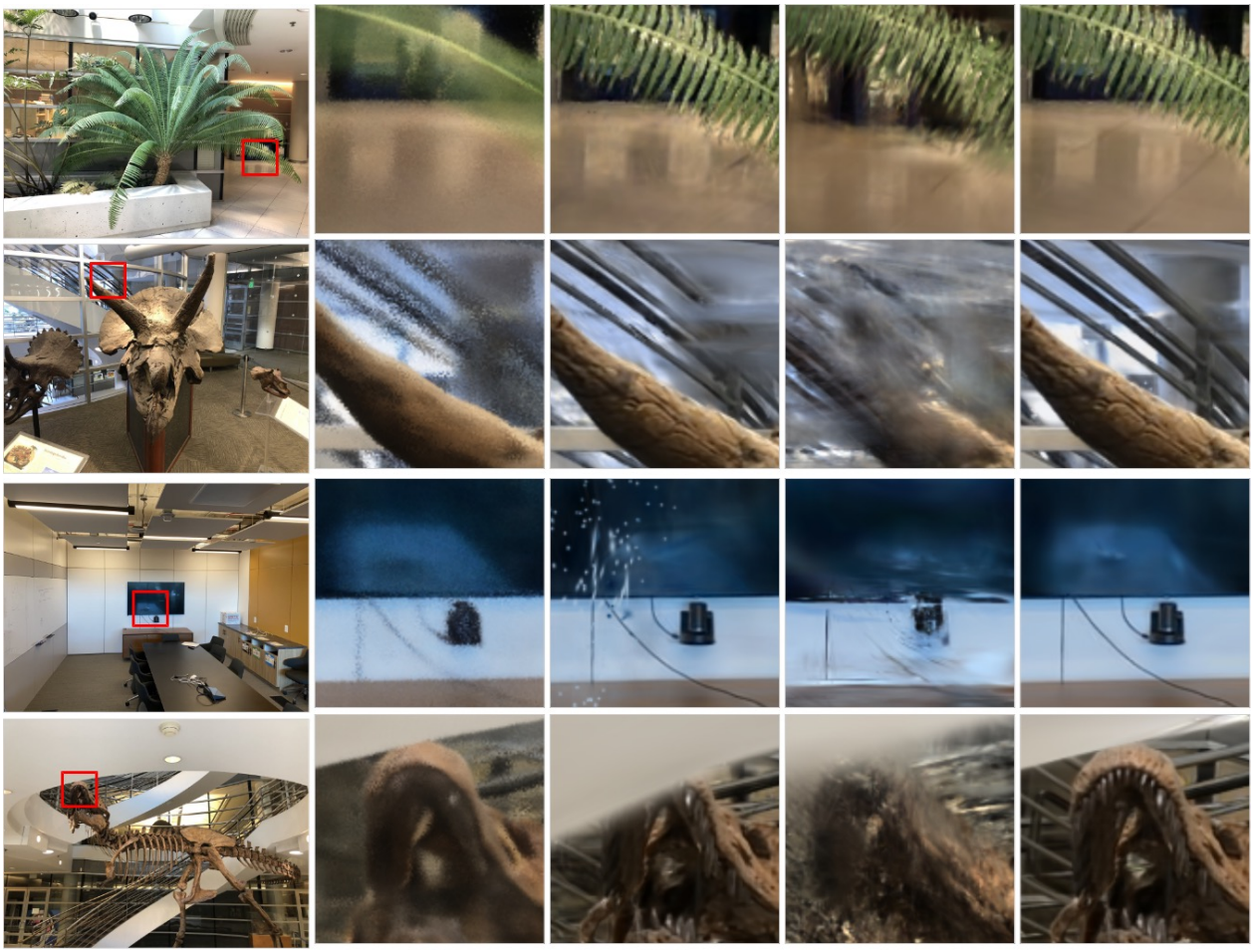

Figure 5. **The qualitative results of novel view synthesis on LLFF forward-facing dataset [41].**

Scenes	COLMAP+3DGS			Ours		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Family	21.92	0.733	0.177	23.03	0.770	0.144
Francis	26.09	0.830	0.214	26.79	0.842	0.186
Ignatius	20.90	0.606	0.281	21.95	0.665	0.234
Train	19.68	0.646	0.277	20.59	0.673	0.252

Table 5. **Quantitative results of novel view synthesis on Tanks-and-Temples dataset**. ↑: higher is better, ↓: lower is better.

model, the convergence of the refinement step can still fail in cases where the derived camera poses are grossly erroneous. We observe that camera pose accuracy is improved with our incremental training pipeline (Ours$_{\text{coarse}}$ *v.s.* Spann3R), which provides much better initial values than Spann3R. See our supplementary for more ablation results.

5. Conclusion

In this paper, we propose ZeroGS to reconstruct neural scenes from unposed images. Our method adopts a pre-trained 3D foundation model as a scene regressor and leverages its learned geometry priors to ease the task of pose-free

Scenes	Spann3R [57]		Ours$_{\text{coarse}}$		Ours$_{\text{refine}}$	
	ΔR	Δt	ΔR	Δt	ΔR	Δt
Fern	39.03	0.767	01.30	0.125	0.26	0.005
Flower	11.91	0.285	16.53	0.609	0.52	0.011
Fortress	08.31	0.152	06.07	0.127	0.04	0.002
Horns	06.98	0.349	14.23	0.145	0.03	0.001
Leaves	44.09	0.801	18.24	0.187	0.22	0.006
Orchids	09.77	0.256	07.22	0.255	0.24	0.006
Room	07.48	0.513	10.22	0.180	0.03	0.001
Trex	32.39	0.758	07.76	0.210	0.03	0.010

Table 6. **Ablation study of camera pose accuracy**.

3DGS training. Based on the learned geometry, we obtain coarse camera poses by RANSAC and PnP solver and refine it with a point-to-camera ray consistency loss. Our training pipeline incrementally registers the image batch into a training buffer and progressively finetunes the model in a self-supervised manner. Our method surpassed state-of-the-art pose-free NeRF/3DGS methods on the LLFF, Mip-NeRF360, and Tanks-and-Temples datasets and comparable or even outperforms 3DGS trained with COLMAP poses.

References

[1] Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Netvlad: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016. 5

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF International Conference on Computer Vision*, pages 5835–5844, 2021. 2, 3

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5460–5469, 2022. 2, 6

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF International Conference on Computer Vision*, pages 19640–19648, 2023. 2

[5] Wenjing Bian, Zirui Wang, Kejie Li, and Jia-Wang Bian. Nope-nerf: Optimising neural radiance field with no pose prior. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4160–4169, 2023. 1

[6] Eric Brachmann and Carsten Rother. Visual camera relocalization from RGB and RGB-D images using DSAC. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(9):5847–5865, 2022. 6

[7] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - differentiable RANSAC for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2492–2500, 2017. 6

[8] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using RGB and poses. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5044–5053, 2023. 4

[9] Eric Brachmann, Jamie Wynn, Shuai Chen, Tommaso Cavallari, Áron Monszpart, Daniyar Turmukhambetov, and Victor Adrian Prisacariu. Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer. *CoRR*, abs/2404.14351, 2024. 2, 4, 5, 6, 7

[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 *IEEE/CVF International Conference on Computer Vision*, pages 14104–14113, 2021. 2

[11] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision - ECCV 2022 - 17th European Conference*, pages 333–350, 2022. 2

[12] Yu Chen and Gim Hee Lee. DBARF: deep bundle-adjusting generalizable neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24–34, 2023. 1, 6, 7

[13] Yu Chen and Gim Hee Lee. Dreg-nerf: Deep registration for neural radiance fields. In *IEEE/CVF International Conference on Computer Vision*, pages 22646–22656, 2023. 2

[14] Yu Chen and Gim Hee Lee. Dogaussian: Distributed-oriented gaussian splatting for large-scale 3d reconstruction via gaussian consensus. *CoRR*, abs/2405.13943, 2024. 3

[15] Yue Chen, Xingyu Chen, Xuan Wang, Qi Zhang, Yu Guo, Ying Shan, and Fei Wang. Local-to-global registration for bundle-adjusting neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8264–8273, 2023. 1

[16] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. GARF: gaussian activated radiance fields for high fidelity reconstruction and pose estimation. *CoRR*, abs/2204.05735, 2022. 1, 2

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. 3

[18] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds, 2024. 2, 3, 6

[19] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5491–5500, 2022. 2

[20] Hongyu Fu, Xin Yu, Lincheng Li, and Li Zhang. CBARF: cascaded bundle-adjusting neural radiance fields from imperfect camera poses. *IEEE Trans. Multim.*, 26:9304–9315, 2024. 1

[21] Yang Fu, Xiaolong Wang, Sifei Liu, Amey Kulkarni, Jan Kautz, and Alexei A. Efros. Colmap-free 3d gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20796–20805, 2024. 1, 3, 6, 7, 2

[22] Lily Goli, Daniel Rebain, Sara Sabour, Animesh Garg, and Andrea Tagliasacchi. nerf2nerf: Pairwise registration of neural radiance fields. In *IEEE International Conference on Robotics and Automation*, pages 9354–9361, 2023. 2

[23] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 684–691, 2004. 2

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

[25] 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*, page 32, 2024. 3

[26] Nishant Jain, Suryansh Kumar, and Luc Van Gool. Robustifying the multi-scale representation of neural radiance fields. *CoRR*, abs/2210.04233, 2022. 2

[27] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18344–18347, 2022. 2

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139:1–139:14, 2023. 1, 3, 4, 6

[29] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Trans. Graph.*, 43(4): 62:1–62:15, 2024. 3

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. 1

[31] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 2, 6

[32] Zhihao Liang, Qi Zhang, Wenbo Hu, Ying Feng, Lei Zhu, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. *CoRR*, abs/2403.11056, 2024. 3

[33] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: bundle-adjusting neural radiance fields. In *IEEE/CVF International Conference on Computer Vision*, pages 5721–5731, 2021. 1, 2, 6, 7

[34] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5166–5175, 2024. 3

[35] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems 33*, 2020. 2

[36] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7814–7823, 2022. 2

[37] Yang Liu, He Guan, Chuanchen Luo, Lue Fan, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. *CoRR*, abs/2404.01133, 2024. 3

[38] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 3, 4

[39] Zhenxing Mi and Dan Xu. Switch-nerf: Learning scene decomposition with mixture of experts for large-scale neural radiance fields. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[40] 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. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow*, pages 405–421. 1, 2, 7

[41] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 38(4):29:1–29:14, 2019. 2, 6, 8

[42] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 2

[43] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. In *Computer Vision - ECCV 2020 - 16th European Conference*, pages 137–154, 2020. 2

[44] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *IEEE/CVF International Conference on Computer Vision*, pages 12159–12168, 2021. 1

[45] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Thomas A. Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12922–12932, 2022. 2

[46] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians, 2024. 4

[47] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 1, 2, 5, 7, 3, 4

[48] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020. 2

[49] Brandon Smart, Chuanxia Zheng, Iro Laina, and Victor Adrian Prisacariu. Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs. *CoRR*, abs/2408.13912, 2024. 2, 3

[50] Xiaowei Song, Jv Zheng, Shiran Yuan, Huan-ang Gao, Jingwei Zhao, Xiang He, Weihao Gu, and Hao Zhao. SA-GS: scale-adaptive gaussian splatting for training-free anti-aliasing. *CoRR*, abs/2403.19615, 2024. 3

[51] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8259–8269, 2022. 2

[52] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020. 2

[53] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben P. Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8238–8248, 2022. 2

[54] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly- throughs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 2

[55] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991. 1

[56] Evangelos Ververas, Rolandos Alexandros Potamias, Jifei Song, Jiankang Deng, and Stefanos Zafeiriou. SAGS: structure-aware 3d gaussian splatting. *CoRR*, abs/2404.19149, 2024. 3

[57] Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *CoRR*, abs/2408.16061, 2024. 3, 4, 6, 7, 8, 1, 2

[58] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 2

[59] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. Dust3r: Geometric 3d vision made easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 2, 3, 4

[60] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields without known camera parameters. *CoRR*, abs/2102.07064, 2021. 2

[61] Yitong Xia, Hao Tang, Radu Timofte, and Luc Van Gool. Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction. *CoRR*, abs/2210.04553, 2022. 2

[62] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8306, 2023. 2

[63] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 3

[64] Botao Ye, Sifei Liu, Haofei Xu, Li Xueting, Marc Pollefeys, Ming-Hsuan Yang, and Peng Songyou. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024. 2

[65] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 1

[66] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *IEEE/CVF International Conference on Computer Vision*, pages 5732–5741, 2021. 2

[67] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 3

ZeroGS: Training 3D Gaussian Splatting from Unposed Images

Supplementary Material

A. Implementation

Neural Scene Regressor. We initialize part of our model with the pretrained weights from Spann3R [57]. Following Spann3R, we use a ViT-large encoder, two ViT-base decoders, and a DPT head [44] for predicting dense pointmaps. We additionally use another DPT head to predict 3D Gaussian primitives. Though Spann3R is trained on images with resolution 224×224, we finetune it on the reconstructed scene with image resolution 512×512 using AdamW [30] optimizer.

Image Registration. We use DSAC to register images and compute coarse camera poses. An image is successfully registered if it has at least $5,000$ inliers with a reprojection threshold of 6 px. We use a number of 64 hypotheses and an inlier alpha of 100. To accelerate registration, the dense pointmaps for each image are downsampled by 4. To further speed up the training and reduce memory footprint, we do not use all pointmaps from the newly registered images to refine camera poses. Instead, we pre-build sparse point tracks $\mathbf{X}_k = \{(\mathbf{u}_{ij}, \mathbf{I}_i)\}$, where $\mathbf{u}_{ij}$ denotes the j-th pixels observed on image $\mathbf{I}_i$. We adopt the Union-Find algorithm to remove duplicate and ambiguous tracks to improve the robustness during refinement. We adopt the Huber loss with a threshold of 0.1 as the robust loss function in Eq. (8).

Run Time and Memory Footprint. Our pipeline converges faster than COLMAP [47]. On the LLFF dataset, our method converges in two epochs, which takes about 25 minutes for each scene; On the MipNeRF360 dataset and the Tanks-and-Temples dataset, our method converges in $5-15$ epochs, which takes about 2 hours for each scene. During training, we evaluate the model and save intermediate results to disks for every $1,000$ iteration. The evaluation time is also included in the training step. Our method takes about 21GB with batch size 1 during training on an NVIDIA 4090 GPU.

Pseudo Algorithm of Our Training Pipeline. We provide the pseudo algorithm of our incremental training pipeline as described in Sec. 3.3 in Alg. 1. At line 1, $\mathcal{V}_i$ denotes the set of graph nodes, and $\mathcal{E}_i$ denotes the set of graph edges. At line 4, $|\cdot|$ denotes the capacity of a set. We align our predicted camera poses to pseudo-ground-truth using the Umeyama [55] algorithm. Note that the camera poses and sparse points of COLMAP are normalized at the end of reconstruction. In line 14, we also normalize our predicted camera poses and dense points before refining the neural scene for fair comparison. We experimentally found this can improve the training stability of 3DGS.

Algorithm 1 Incremental Neural Reconstruction Algorithm

Require: a set of (unordered) images $\{\mathbf{I}_i\}$, maximum iteration per epoch iter_{max}
Ensure: Camera poses $\{\mathbf{T}_k\}$, 3D Gaussian primitives $\{\mathbf{G}_k\}$

1: Construct a similarity graph $\mathcal{G}_{\text{sim}} = (\mathcal{V}_i, \mathcal{E}_i)$
2: Initialization from a seed image $\mathbf{I}_{\text{seed}}$ (*cf*. Sec. 3.3.1)
3: Registered image set $(\mathcal{I}_{\text{reg}}, \mathcal{T}_{\text{reg}}) = \{(\mathbf{I}_{\text{seed}}, \mathbf{T}_{\text{seed}})\}$
4: **while** $|\mathcal{I}_{\text{reg}}| < |\{\mathbf{I}_i\}|$ **do**
5: Register a new images $(\mathcal{I}_{\text{new}}, \mathcal{T}_{\text{new}})$ by Eq. (7)
6: Refine newly registered camera poses by Eq. (8)
7: Update training buffer using $(\mathcal{I}_{\text{new}}, \mathcal{T}_{\text{new}})$
8: iter $:= 0$
9: **while** iter $<$ iter_{max} **do**
10: Finetune scene regressor f_{SCR} using Eq. (6)
11: iter $:=$ iter $+ 1$
12: $\mathcal{I}_{\text{reg}} := \mathcal{I}_{\text{reg}} + \mathcal{I}_{\text{new}},\ \mathcal{T}_{\text{reg}} := \mathcal{T}_{\text{reg}} + \mathcal{T}_{\text{new}}$
13: Finalize camera poses $\mathcal{T}_{\text{reg}} = \{\mathbf{T}_k\}$ using (8)
14: Normalize camera poses $\mathcal{T}_{\text{reg}}^{\text{norm}} = \text{Normalize}(\mathcal{T}_{\text{reg}})$
15: Finalize neural scene $\{\mathbf{G}_k\}$ (*cf*. Sec. 3.3.3)

B. Additional Results

Ablation of Pose Refinement. We present the visual comparison of our method with ($\text{Ours}_{\text{refine}}$) and without ($\text{Ours}_{\text{coarse}}$) the refinement step in Fig. 6. As is shown in Fig. 6, after camera pose refinement, camera poses are aligned closer to ground truth. Compared to the camera poses obtained from Spann3R of the first row in Fig. 3 and the third row in Fig. 7, the coarse camera poses are closer to the ground truth, which is coherent with the quantitative results provided in Table 6.

Ablation of Finalizing Reconstruction. We ablate the effectiveness of finalizing the camera poses in our training pipeline in Table 7. The unit for rotation error is degree. We denote our method without the finalizing step as Ours_{nf}. We can see that the finalization effectively mitigates the error accumulation in both camera rotations and translations. We also emphasize that the camera pose finalizing step is important to the neural scene refinement step since 3DGS is sensitive to even small perturbance camera poses. Moreover, jointly optimizing explicit 3DGS and camera poses during training has limited effect when camera poses are close to ground truth and can even diverge the training [65].

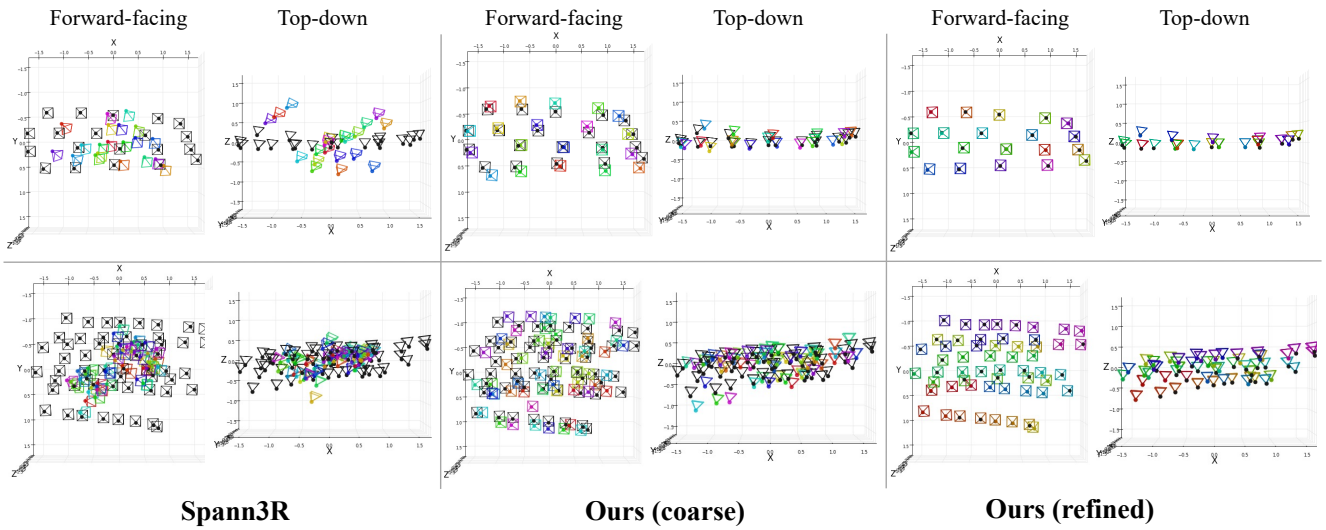

Figure 6. **Ablation of camera poses refinement on the LLFF dataset** (Zoom in for best view). Top and bottom are respectively the camera poses of the 'Fern' and 'Room' scenes.

Scenes	Bicycle		Counter		Garden		Kitchen	
	ΔR	Δt	ΔR	Δt	ΔR	Δt	ΔR	Δt
Ours$_{nf}$	0.096	0.015	0.041	0.007	0.095	0.012	0.150	0.219
Ours	0.035	0.005	0.029	0.002	0.028	0.002	0.052	0.008

Table 7. **Quantitative results of camera pose accuracy on MipNeRF360 dataset**. Red color denotes the best results.

More Results of Camera Poses. We include the quantitative results of the camera pose accuracy evaluated on the Tanks-and-Temples dataset in Table 8. The visual comparison is also provided in Fig. 9. More visualization results of the aligned camera pose for the LLFF dataset and the MipNeRF360 dataset are respectively provided in Fig. 7 and Fig. 8. We can observe that CF-3DGS [21] failed to produce faithfully camera poses on the LLFF dataset, which has been analyzed in the main paper. While ACE0 [9] performs very well on the MipNeRF360 dataset, the training is unstable even with a fixed seed number, hence we can not reproduce the comparable result of ACE0 on the Garden scene (*cf*. Table 3) of the MipNeRF360 dataset. Moreover, we find that ACE0 performs poorly on the LLFF dataset. This may be due to the MLP decoder of ACE0 maps individual pixels to 3D space, while it requires well-distributed training views to constrain the network. However, the LLFF dataset contains only forward-facing cameras, which do not provide strong constraints from different view directions and therefore degenerate the training of ACE0.

More Qualitative Results of Novel View Synthesis. We present more qualitative results of the MipNeRF360 dataset and the Tanks-and-Temples dataset respectively in Fig. 10 and Fig. 11 on the novel view synthesis task. The quantitative camera pose accuracy only reveals how close the predicted camera poses to the COLMAP poses. However, it cannot distinguish which one is more accurate since COLMAP poses are only pseudo-ground-truth and it can produce erroneous camera poses. Nonetheless, the results of novel view synthesis provide better metrics to show which one is better when two camera poses are close. In Fig. 10 and Fig. 11, we can observe that our method can render finer details when we zoom into the same areas. The visual comparison also provides coherent support to the quantitative results of novel view synthesis in Table 4 and Table 5. More reconstruction results of camera pose and pointmaps are included in Fig. 12.

Scenes	Spann3R [57]		Ours	
	ΔR	Δt	ΔR	Δt
Family	16.98	0.378	0.036	0.003
Francis	14.19	0.361	0.030	0.002
Ignatius	11.23	0.313	0.028	0.002
Train	17.33	0.286	0.065	0.011

Table 8. **Quantitative results of camera pose accuracy on Tanks-and-Temples dataset**. Red color denotes the best results.

C. More Discussion

Limitations. Though our method can produce higher-quality reconstruction results in terms of both camera poses and novel view synthesis, it requires more GPU memory and training time than ACE0 since we are finetuning transformers, which limits its application on larger scenes. In addition, the training convergence speed of our method relies on the pretrained model of Spann3R. Note that DUSt3R is trained on a mixture of 8 datasets, while Spann3R is only trained on the subset of these datasets.

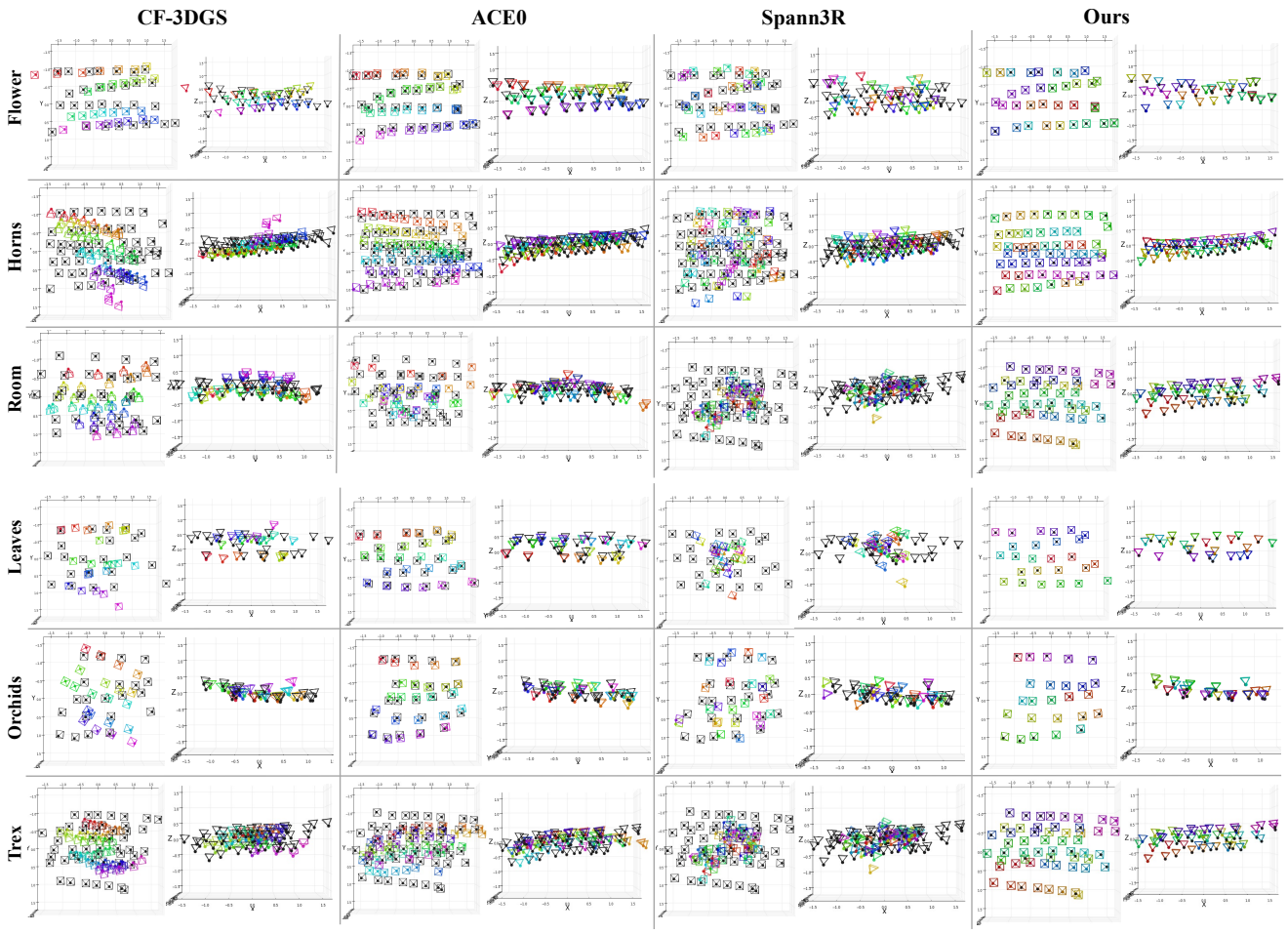

Figure 7. More **qualitative comparisons of camera poses accuracy on the LLFF dataset** (Zoom in for best view). Black: pseudo-ground-truth camera poses obtained from COLMAP [47]. Colored: predicted camera poses.

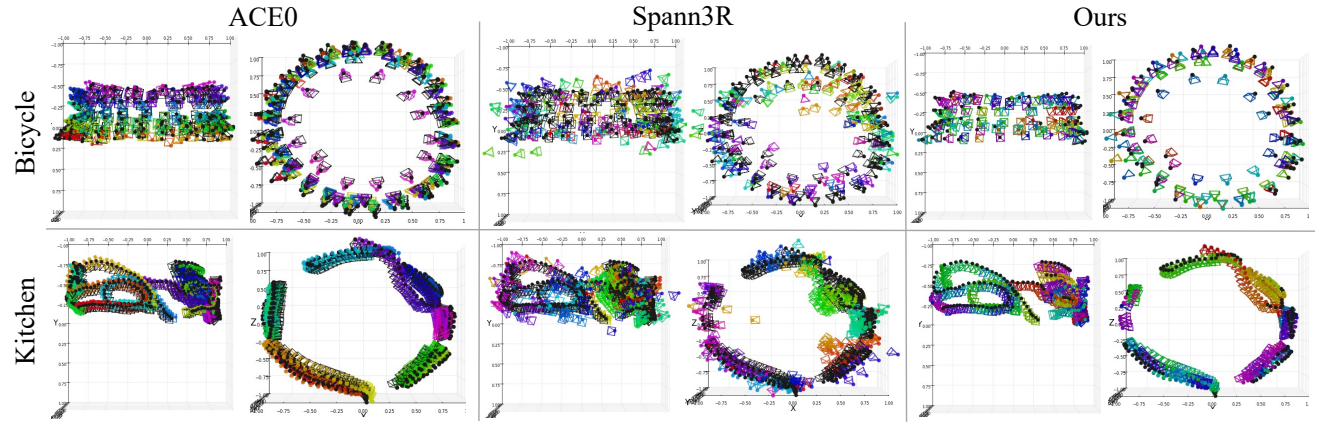

Figure 8. More **qualitative comparisons of camera poses accuracy on the MipNeRF360 dataset** (Zoom in for best view). Black: pseudo-ground-truth camera poses obtained from COLMAP [47]. Colored: predicted camera poses.

Future Work. Our future work includes distilling the pre-trained large foundation model into a lightweight network to speed up the training and reduce the GPU memory requirement during training. We will also explore the applicability of the light-weight model on larger-scale and more diverse scenes.

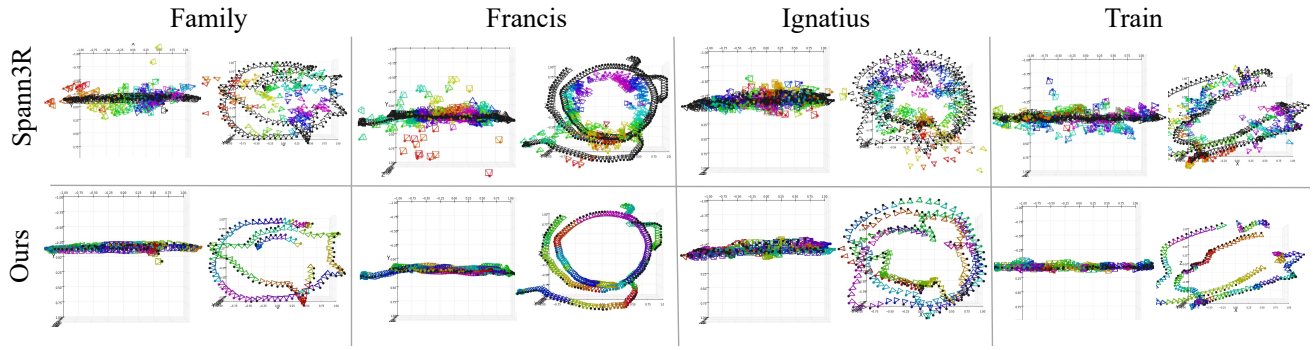

Figure 9. More **qualitative comparisons of camera poses accuracy on the Tanks-and-Temples dataset** (Zoom in for best view). Black: pseudo-ground-truth camera poses obtained from COLMAP [47]. Colored: predicted camera poses.

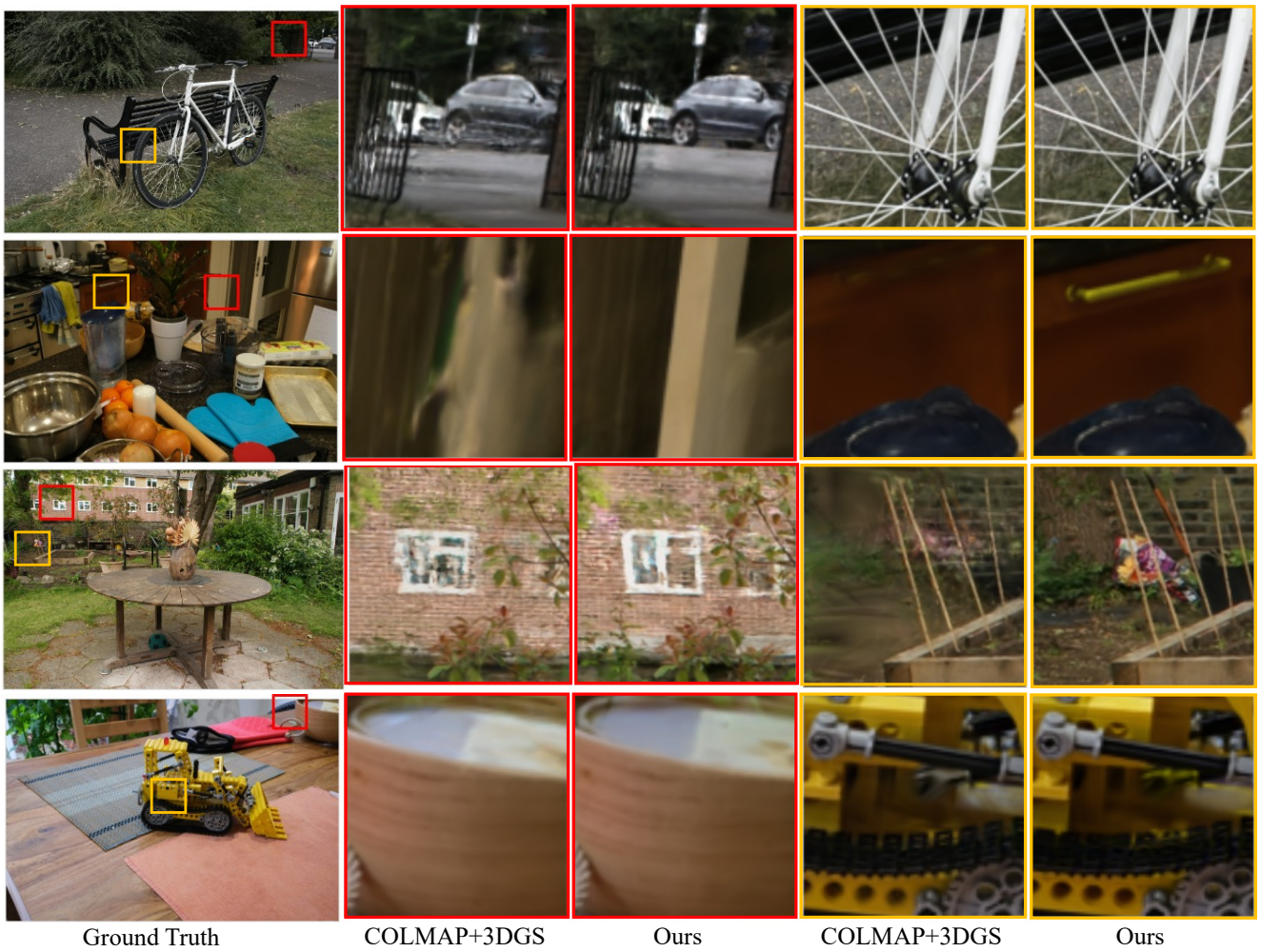

Figure 10. More **qualitative comparisons of novel view synthesis on the MipNeRF360 dataset**. From top to bottom are respectively the results on scenes of bicycle, counter, garden, and kitchen.

Figure 11. More **qualitative comparisons of novel view synthesis on the Tanks-and-Temples dataset**. From top to bottom are respectively the results on scenes of the family, Francis, Ignatius, and the train.

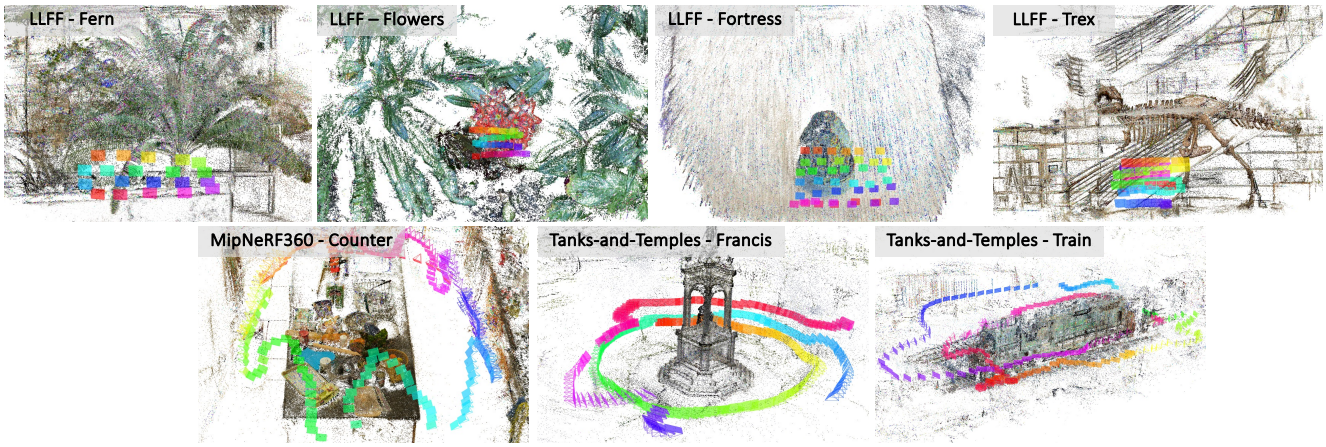

Figure 12. More **visual reconstruction results** on real-world datasets.