Global Latent Neural Rendering

Thomas Tanay Matteo Maggioni
Huawei Noah's Ark Lab
thomas-tanay.github.io/convglr

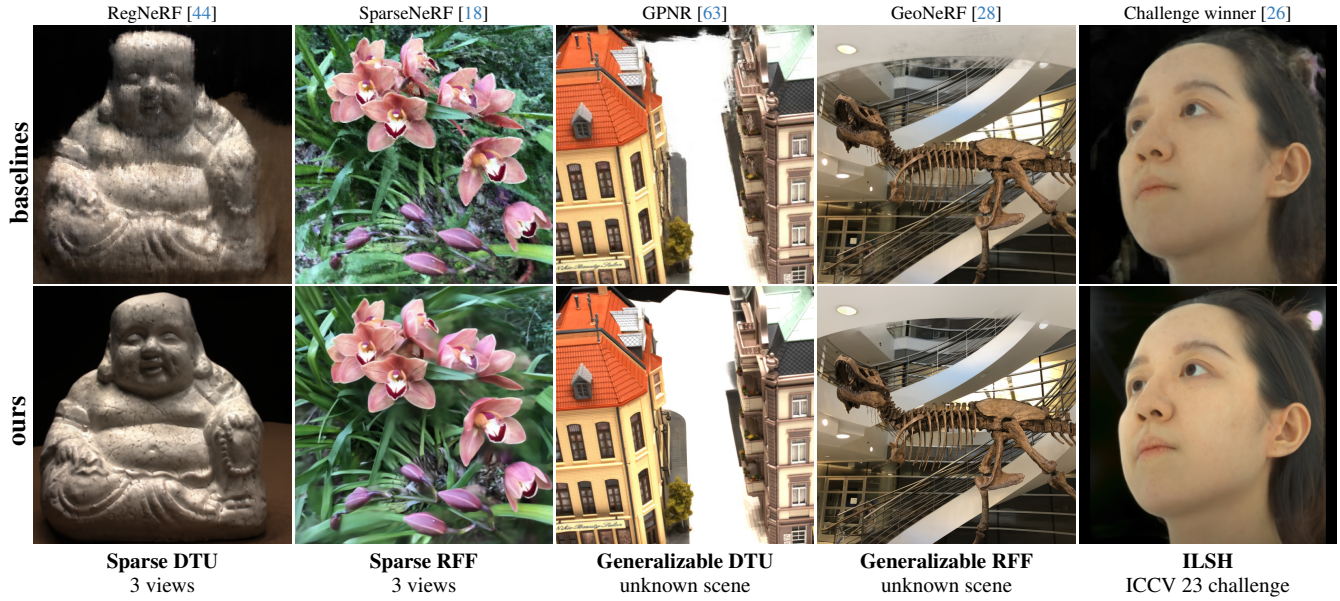

Figure 1. Qualitative comparison of our method with various baselines under 5 different experimental setups. Our method renders target views in a low-resolution latent space and operates over all camera rays jointly. It produces significantly better geometries and textures than previous sparse and generalizable methods, which render light rays independently and typically suffer from grainy artifacts.

Abstract

A recent trend among generalizable novel view synthesis methods is to learn a rendering operator acting over single camera rays. This approach is promising because it removes the need for explicit volumetric rendering, but it effectively treats target images as collections of independent pixels. Here, we propose to learn a global rendering operator acting over all camera rays jointly. We show that the right representation to enable such rendering is the 5-dimensional plane sweep volume, consisting of the projection of the input images on a set of planes facing the target camera. Based on this understanding, we introduce our Convolutional Global Latent Renderer (ConvGLR), an efficient convolutional architecture that performs the rendering operation globally in a low-resolution latent space. Experiments on various datasets under sparse and generalizable setups show that our approach consistently outperforms existing methods by significant margins.

1. Introduction

Significant progress has been made on novel view synthesis in recent years, both in terms of image quality and rendering speed [2, 3, 6, 15, 30, 42, 43]. However, a lot of this progress has focused on the scene-specific formulation of the problem, where models are trained to fit one scene. We are interested here in the generalizable formulation, where novel views of unknown scenes can be rendered directly from a set of posed input views [5, 41, 63, 70, 76].

This generalizable formulation is challenging because it requires to reason about the geometry of the scene for each target image, instead of solving the geometry problem as a preliminary step. It also typically relies on a much sparser number of input views (3 to 16 here) while the scene-specific formulation routinely uses 100s of input views. However, we believe that it is ultimately more powerful because 1) sparse setups are common in real world applications [11, 18, 44] and 2) it provides the ability to

reason about unkown environments and could pave the way for the training of large scale 3D vision models [12]. Most recent works on generalizable novel view synthesis learn to predict 5D radiance fields based on some form of geometric reasoning before applying volumetric rendering; a fixed operation consisting in integrating the radiance over light rays [5, 28, 70, 76]. A recent development is to use a 4D light field approach and predict the color of camera rays directly, effectively learning the rendering operation itself [12, 63, 64]. This later approach is promising because it removes the need for explicit volumetric rendering but so far, it is still implemented on a single-ray basis.

In this work, we adopt a 4D light field approach and learn a global rendering operator acting over all camera rays jointly. We achieve this by revisiting plane sweep volumes (PSVs), obtained by projecting the input views on a set of planes distributed parallel to the target image plane. In particular, we observe that PSVs implicitly encode the epipolar geometry of the scene such that mixing information *across epipolar lines* can be implemented with operations along the view dimension of PSVs, mixing information *along epipolar lines* can be implemented with operations along the depth dimension of PSVs and mixing information *between light rays* can be implemented with operations along the height and width dimensions of PSVs. Based on this understanding, we introduce a Convolutional Global Latent Renderer (ConvGLR), an efficient convolutional architecture that renders novel views directly from plane sweep volumes. ConvGLR is a 4 step model that 1) arranges the PSV into groups of successive depths, 2) aggregates information across views in a depth-independent manner while reducing the spatial dimension of the representation, 3) performs global latent rendering by progressively collapsing the depth dimension and 4) upsamples the rendered representation into a final output. This design is validated in mutliple experiments on the DTU [27], Real-Forward Facing [42] and Spaces [14] datasets under established sparse and generalizable setups [5, 42, 44], as well as on the recently introduced ILSH dataset [77] in the context of a public novel view synthesis challenge with held-out test views [25, 26]. Our main contributions are as follow:

- We introduce *global latent neural rendering*, a simple and powerful approach to novel view synthesis that consists in learning a generalizable light field model from plane sweep volumes.
- We design a *Convolutional Global Latent Renderer* (ConvGLR), a convolutional architecture that implements global latent neural rendering efficiently.
- We evaluate ConvGLR extensively on sparse and generalizable setups as well as on a public novel view synthesis challenge with held-out test views, and significantly outperform existing methods in all cases.

2. Related work

NeRFs Neural Radiance Fields [2, 3, 42] model the 5D radiance and 3D density fields of individual scenes in the weights of an MLP. They have become highly popular for their ability to produce high quality renderings of complex scenes from arbitrary viewpoints. They tend to be relatively slow at rendering time, although significant speed-ups have been obtained by removing the neural representation entirely [15], using multiresolution hash encodings [43], tensor decompositions [6] or 3D gaussians [30]. NeRF models also struggle on scenes that are viewed under very sparse conditions. Multiple attempts have been made at addressing this limitation, often by training on missing views using auxiliary losses. For instance, DietNeRF [24] uses a semantic consistency loss based on the CLIP vision transformer [48]. RegNeRF [44] uses appearance and geometry regularization based on a normalizing flow model and a smoothness loss. FlipNeRF [55] increases the number of training rays by reflecting the existing ones and introduces two new regularization losses. MixNeRF [56] models rays with mixture densities and introduces depth estimation as proxy objective. DSNeRF [11] exploits readily-available depth supervision signals obtained from COLMAP [54]. SparseNeRF [18] improves the use of depth maps further by introducing a depth ranking constraint. Similarly to our approach, GANeRF [52] improves the rendering operation on training views by acting on groups of pixels via an adversarial loss applied on patches. However, this is in the context of a scene-specific model that still relies on fixed volumetric rendering over individual camera rays.

Light fields In free space, the radiance is constant over light rays and scenes can be encoded as 4D light fields. This idea has been used in early works to perform novel view synthesis without [32], or with limited [16] geometric reasoning by relying on a dense sampling of the scene. Recent methods have focused on sparser setups in a learning based way [1, 29, 60, 63, 64], often with a focus on modeling non-Lambertian effects [1, 64]. An important distinction between these works and neural radiance fields is that they learn the rendering operation instead of relying on classival volumetric rendering. Contrary to our method, however, they still learn the rendering operation over single light rays.

Implicit geometry A popular approach to novel view synthesis is to reason about the geometry of the scene implicitly [58], typically via known epipolar constraints. For instance, GRF [67] and PixelNeRF [76] extract image features along epipolar lines to encode 3D points, and render camera rays using volumetric rendering. IBRNet [70] and NerFormer [49] follow a similar approach while using more sophisticated transformer-based architectures. DynIBaR [34] extends epipolar line sampling in a motion-aware fashion. LFNR [64] and GPNR [63] also process

image patches extracted along epipolar lines with transformers, while using a 4D light field model predicting the color of individual camera rays directly. Finally, the method from [12] extends this approach to the challenging scenario of wide-baseline stereo pairs. Our method also uses implicit geometric reasoning, but it does so with plane sweep volumes which are richer epipolar encodings than simple epipolar lines.

Explicit geometry In contrast with the previous category, a number of novel view synthesis methods rely on explicit geometric modeling of the scene [58]. Early methods included 3D warping based on depth information [40], layered depth images to deal with occlusions [57] or view-dependent texture maps inspired from computer graphics [10]. More recent methods still rely on depth maps [11, 18, 45, 47] or rely on the construction of a geometric scaffold or mesh [8, 21, 50, 51] However, these methods are vulnerable to inacuracies in the estimation of the underlying geometry. In contrast, our method does not use any form of explicit geometric reasoning.

Layered representations The plane sweep algorithm was introduced in the context of multi-view stereo in [9] and was first applied to novel view synthesis using a layered representation in [65]. The term *plane sweep volume* (PSV) is often used to refer to the 4D tensor obtained by projecting *one* input image on a set of depth planes facing a target or reference camera [13, 14, 41, 78]. With the advent of deep learning, several methods have been introduced to perform generalizable novel view synthesis by processing PSVs. Early methods typically produced layered representations that consisted in a mix of depth maps, oclusion maps and color maps [13, 29, 45]. Later methods focused on the multiplane image representation (MPI), which consists in a set of RGBα images that can be projected to novel viewpoints and rendered using alpha blending [14, 41, 62, 78]. MPIs have also been used in a scene-specific manner [71] and to generate novel views from a single image [19, 33, 68]. Layered depth images are MPI variants where an extra depth channel is predicted [22, 31, 37, 57, 61]. Finally, multiplane feature representations were recently introduced for multi-frame denoising [66]. Our method differs from these works in one important way: instead of producing a layered representation that is rendered through summation or alpha blending, it learns the rendering operation in a low-dimensional latent space.

3D cost volumes A variant of the plane sweep algorithm consists in extracting deep features from the input images indepedently, constructing plane sweep volumes from the deep features, and computing the variance over the input views [74]. Such 3D cost volumes have been used extensively in the literature on multi-view stereo (MVS) [7, 17, 23, 72, 73, 75], and have recently been combined with NeRFs for novel view synthesis [5, 28, 36, 39]. MVSNeRF [5] in particular computes a cost volume centered on the reference view, refines it with a 3D CNN, predicts radiance and density fields using an MLP and finally integrates over camera rays using volumetric rendering. GeoNeRF [28] instead computes cascaded cost volumes centered on the input views, refines these cost volumes using multi-head attention, and again predicts radiance and density fields using MLPs before integrating over camera rays. Our methods differs from these works in three ways: it uses a PSV representation instead of a cost volume, it learns the rendering operation instead of applying fixed volumetric rendering, and it renders all the camera rays jointly instead of independently.

3. Background

Consider a set of V *input views* of a scene, consisting of color images and camera parameters. The images are of height H and width W, with red-green-blue color channels, and can be stacked into a 4D tensor $\boldsymbol{I} \in \mathbb{R}^{V\times 3\times H\times W}$. The camera parameters $\boldsymbol{P}$ consist of an intrinsic tensor $\boldsymbol{K} \in \mathbb{R}^{V\times 3\times 3}$ and an extrinsic tensor that can be split into a rotation tensor $\boldsymbol{R} \in \mathbb{R}^{V\times 3\times 3}$ and a translation tensor $\boldsymbol{t} \in \mathbb{R}^{V\times 3\times 1}$. Now consider a distinct *target view* with ground-truth image $\boldsymbol{I}_*$, and camera parameters $\boldsymbol{P}_* = \{\boldsymbol{K}_*, \boldsymbol{R}_*, \boldsymbol{t}_*\}$. We are interested in *novel view synthesis*, which consists in predicting an estimate $\tilde{\boldsymbol{I}}_*$ of the target image $\boldsymbol{I}_*$, given the input images $\boldsymbol{I}$, the input camera parameters $\boldsymbol{P}$ and the target camera parameters $\boldsymbol{P}_*$.

There exists two main formulations of this problem. The first one learns a *scene-specific* function $\mathcal{F}_{\boldsymbol{I},\boldsymbol{P}}$ on the input views, such that novel views can be rendered from novel camera parameters:

$$\tilde{\boldsymbol{I}}_* = \mathcal{F}_{\boldsymbol{I},\boldsymbol{P}}(\boldsymbol{P}_*)\,. \tag{1}$$

The function $\mathcal{F}_{\boldsymbol{I},\boldsymbol{P}}$ is trained on views from a single predefined scene, and can be used to render novel views for that scene only. The second formulation learns a *scene-agnostic* or *generalizable* function $\mathcal{F}$ on sets of input views and target camera parameters, such that novel views can be rendered from novel sets of input views and target camera parameters:

$$\tilde{\boldsymbol{I}}_* = \mathcal{F}(\boldsymbol{I}, \boldsymbol{P}, \boldsymbol{P}_*)\,. \tag{2}$$

This time, the function $\mathcal{F}$ is trained on a large corpus of scenes, and can be used to render novel views from scenes that have not been seen during training.

The scene-specific formulation is a defining characteristic of NeRF [42] and its extensions [2, 3, 6, 43] which model the function $\mathcal{F}_{\boldsymbol{I},\boldsymbol{P}}$ indirectly through two fields: a radiance field returning a color for every point in space and viewing direction (5D$\rightarrow$ 3D function) and a density field

returning a density for every point in space (3D→1D function). The target image $\tilde{\boldsymbol{I}}_*$ is then rendered by integrating the two fields over camera rays using classical volumetric rendering. For scenes that mostly consist of free space (as is often the case), the 5D radiance field model is redundant because the radiance remains constant along light rays. Light field networks [1, 60, 64] rely on this observation to directly model the function $\mathcal{F}_{\boldsymbol{I},\boldsymbol{P}}$ as a light field returning a color for every light ray (4D→3D function).

Among generalizable methods, a well-known family are the models that predict multiplane image representations [14, 41, 62, 78]. They typically process plane sweep volumes and predict a 3D radiance field with no view dependence (in their standard form) and a 3D density field as a discrete set of RGBα images, that are rendered through alpha-blending. Generalizable neural radiance fields learn a NeRF model on top of a geometric representation, which can rely on 2D deep features extracted along epipolar lines [67, 70, 76] or 3D cost volumes [5, 28, 36, 39]. Existing generalizable light field networks [12, 63] also extract image patches or features along epipolar lines, but they learn the rendering operation and directly predict a pixel color. In this work, we introduce a generalizable light field model that learns to render images globally, by operating over all the camera rays jointly in a low-resolution latent space. We summarize the difference between our approach and various previous methods in Table 1.

methods	formulation	model	rendering
Multi-plane images [14, 41, 62, 78]	**generalizable**	3D radiance field + 3D density field	fixed pointwise
Neural radiance fields [2, 3, 6, 42, 43]	scene-specific	5D radiance field + 3D density field	fixed pointwise
Generalizable neural radiance fields [5, 28, 39, 67, 70, 76]	**generalizable**	5D radiance field + 3D density field	fixed pointwise
Light field networks [1, 60, 64]	scene-specific	**4D light field**	none / learned pointwise
Generalizable light field networks [12, 63]	**generalizable**	**4D light field**	**learned** pointwise
Convolutional global latent rendering (ours)	**generalizable**	**4D light field**	**learned global**

Table 1. **Taxonomy of novel view synthesis approaches.** We distinguish methods according to the formulation they use (scene-specific vs generalizable), the model they learn (radiance field + density field vs light field) and the type of rendering they apply (fixed vs learned and pointwise vs global).

4. Method

We first define the Plane Sweep Volume (PSV) and highlight some of its interesting properties. We then introduce global latent neural rendering, a new generalizable approach to novel view synthesis, and our Convolutional Global Latent Renderer (ConvGLR), an efficient implementation of it. Finally we discuss some implementation details.

4.1. The Plane Sweep Volume

Consider a set of D depth planes distributed parallel to the target image plane $\boldsymbol{I}_*$ such that they share the same normal $\boldsymbol{n}_*$. The depth planes are uniquely defined by their distances $\{a_d\}_{d=1}^D$ from the target camera center and these distances are assumed to be chosen such that the scene of interest is adequately covered (we discuss the choice of these distances in practice in Sec. 4.4). We define the plane sweep volume (PSV) as the 5D tensor $\boldsymbol{X} \in \mathbb{R}^{D\times V\times 3\times H\times W}$, obtained by projecting each input image $\boldsymbol{I}_v$ on each of the D depth planes.[1] Formally, each projected image $\boldsymbol{X}_{dv}$ is obtained by applying a homography to $\boldsymbol{I}_v$, represented by a 3×3 matrix $\boldsymbol{H}_{dv}$. Assuming without loss of generality that the world origin is at the target camera center such that $\boldsymbol{R}_*$ is the identity, $\boldsymbol{t}_* = \boldsymbol{0}$ and $\boldsymbol{n}_* = (0,0,1)^\top$, each homography matrix is defined as [20]:

$$\boldsymbol{H}_{dv} = \boldsymbol{K}_v \left(\boldsymbol{R}_v - \frac{\boldsymbol{t}_v \, \boldsymbol{n}_*{}^\top}{a_d} \right) \boldsymbol{K}_*^{-1} . \tag{3}$$

The plane sweep volume is a highly structured tensor that encodes the epipolar geometry between the input views and the target view [9, 13, 65]. Indeed, consider the camera ray passing through a pixel location (h, w) in the target image plane. This camera ray projects as a set of epipolar lines in the input views. Then by construction, the PSV slice:

$$\boldsymbol{r}_{hw} = \{\{\{\boldsymbol{X}_{dvchw}\}_{c=1}^3\}_{d=1}^D\}_{v=1}^V \tag{4}$$

contains pixels sampled along these epipolar lines at matching depths (see Fig. 2). In other words, $\boldsymbol{r}_{hw}$ can be seen as an encoding of the camera ray passing through (h, w), given the input views. This is particularly useful, because adjacent camera rays have adjacent encodings in the PSV and can be processed together using simple local operators. More precisely, the PSV is structured such that 1) operations along the depth dimension are operations along individual epipolar lines, 2) operations along the view dimension are operations between corresponding epipolar lines and 3) operations along the height and width dimensions are operations between nearby camera rays.

4.2. Global Latent Neural Rendering

We propose a simple and powerful novel view synthesis approach that consists in learning a generalizable light field model $\mathcal{F}$, directly from plane sweep volumes $\boldsymbol{X}$:

$$\tilde{\boldsymbol{I}}_* = \mathcal{F}(\boldsymbol{X}) \tag{5}$$

where $\mathcal{F}$ is implemented as a convolutional neural network. This approach fundamentally differs from the recent line of

[1] The term *plane sweep volume* is often used to refer to 4D tensors obtained by projecting *one* input view on the depth planes. The definition we use here generalizes this to more views.

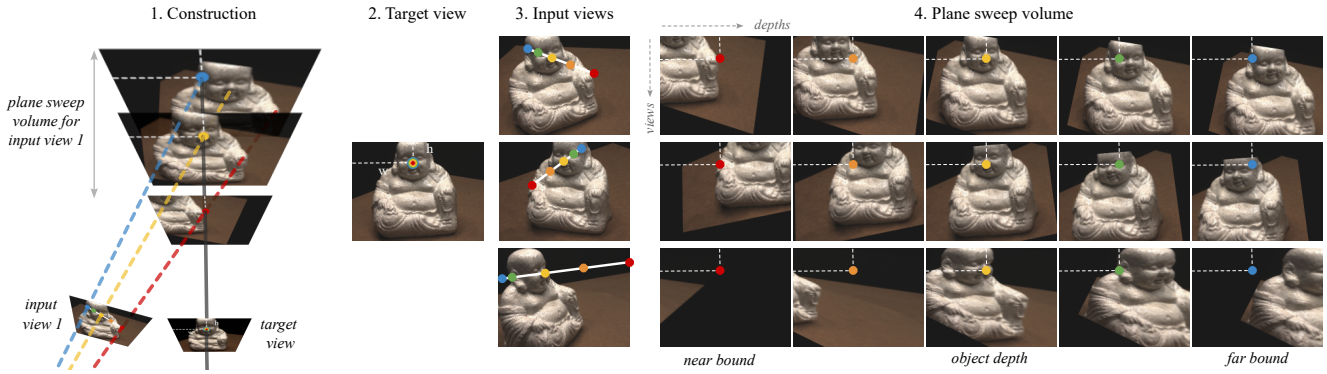

Figure 2. **The epipolar geometry of the plane sweep volume.** 1. The PSV is constructed by projecting each input view on a set of planes distributed parallel to the target image plane. 2. The camera ray passing through the pixel location (h, w) in the target image plane (gray line in 1.) projects as a set of epipolar lines in the input views (white lines in 3.). 4. Moving along the depth dimension of the PSV at pixel location (h, w) is equivalent to moving along the corresponding epipolar lines for each input view. The actual depth of the object at pixel location (h, w) is found when the local image features match across views (yellow dot).

works that use transformers to process image patches extracted along epipolar lines [12, 34, 39, 63, 64, 70], because it uses the plane sweep volume to organise the computation and allows to process camera rays jointly. It also differs from the line of works on layered representations and multiplane images [13, 14, 29, 41, 45, 78], because it learns the rendering operation, instead of keeping the depths separated and relying on alpha-compositing.

The main challenge faced by our proposed approach is the size of the PSV: a 5D tensor $\boldsymbol{X} \in \mathbb{R}^{D\times V\times 3\times H\times W}$ needs to be processed efficiently using convolutions to produce a 3D rendered image $\tilde{\boldsymbol{I}}_* \in \mathbb{R}^{3\times H\times W}$. Our solution is illustrated in Figure 3 and has the following structure (see the Supp. Mat. for more details).

Grouped PSV Similarly to the literature on multiplane representations [14, 41], the 5D PSV is treated as a 4D tensor of shape $D\times 3V\times H\times W$, such that the input views are processed together from the very first layer of the network. We show in our ablation study (see Tab. 7) that this approach is more powerful that the alternative one that constructs a 3D cost-volume [5, 28, 36, 39]. We then view the PSV as a tensor of shape $\frac{D}{G}\times 3GV\times H\times W$ for a group size G. This step significantly reduces the computational load by allowing to process the depths in groups, and effectively reduces the number of depths from D to $D_G = \frac{D}{G}$.

Multi-view matching Early layers aggregate information accross views, and treat the D_G depths independently from each other by keeping them in the batch dimension. The spatial resolution is reduced $4\times$, alternatively using 2D convolutions with stride 2 and 2D resblocks, following a typical encoder-decoder or Unet [53] structure. The number of channels at the base of the network is a hyperparameter C, and the channels are doubled after each spatial downsampling. This block results in a latent volumetric representation $\boldsymbol{Y} \in \mathbb{R}^{D_G\times 4C\times \frac{H}{4}\times \frac{W}{4}}$.

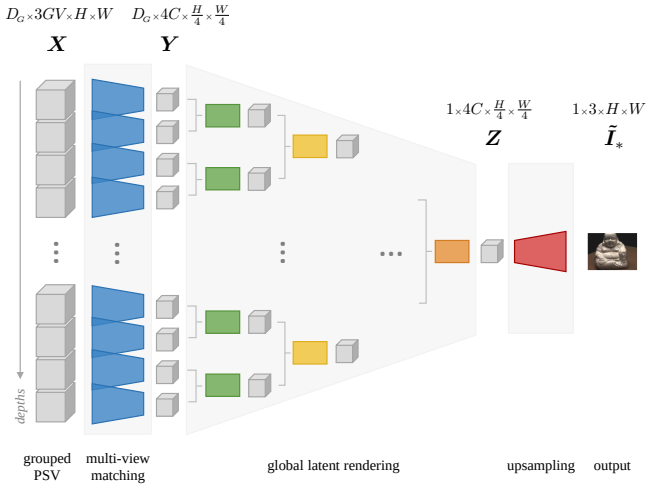

Figure 3. **Overview of ConvGLR.** The 4D grouped PSV $\boldsymbol{X}$ is turned into a latent volumetric representation $\boldsymbol{Y}$, then rendered into a latent novel view $\boldsymbol{Z}$ and finally upsampled into the novel view $\tilde{\boldsymbol{I}}_*$. All the colored blocks are implemented with 2D convolutions and resblocks. Blocks with matching colors share weights.

Global latent rendering The rendering operation is fundamentally an integration over the depth dimension, and consists in reducing the depth of the latent tensor $\boldsymbol{Y}$ to 1. We implement it by iteratively grouping the depths by pairs and processing them with 2D resblocks. This emulates the use of 3D resblocks with a kernel size of 2 and a stride of 2 along the depth dimension, without requiring memory-expensive transpose operations. This block produces a globally rendered latent representation $\boldsymbol{Z} \in \mathbb{R}^{1\times 4C\times \frac{H}{4}\times \frac{W}{4}}$.

Upsampling Finally, the output $\tilde{\boldsymbol{I}}_*$ is produced by upsampling the latent representation $4\times$, alternatively using $2\times$

bilinear interpolation and 2D resblocks, as is typically done in the super-resolution literature [4, 35].

4.3. Additional conditioning

While the PSV is an information-rich encoding of the input views, we propose to augment it further with two additional conditional inputs. We show in our ablation study (see Tab. 7) that these two conditional inputs have a negligible negative impact on the computational load, but have a significant positive impact on performance.

Positional encoding First, we concatenate to the PSV the spatial coordinates (h, w) in the form of two extra channels normalized in the $[0, 1]$ range. We do not use any Fourrier encoding to avoid overloading an already large PSV tensor. Explicitly feeding the spatial coordinates is a simple way to make the model *spatially-adaptive* [38], such that it renders specific groups of pixels differently depending on their location in the image (e.g. outer pixels are more likely to belong to the background, and be of specific colors). This use of positional encoding is closer to its original use in transformers [69], where it was introduced as a way of injecting information about the position of tokens in a sequence, than its use in NeRF [42], where it helps encode high-frequency content.

Angular encoding Let $\boldsymbol{u}_{dv}$ be the unit vector pointing in the direction between the camera center of view v and the center of the depth plane d. Remembering that the normal to the target image plane is $\boldsymbol{n}_*$, we concatenate the dot product $\boldsymbol{u}_{dv} \cdot \boldsymbol{n}_*$ as an additional channel to each projected image $\boldsymbol{X}_{dv}$ in the PSV. The motivation is two-fold. First, this dot product measures an angular distance between the target view and view v (as seen from the depth plane d), and hence, it is a good measure of the similarity between the two views at that depth. Second, we hypothesise that this can help model finegrained view-dependent effects, by making the input more explicitly view-dependent.[2]

4.4. Implementation details

Similarly to other novel view synthesis methods, the *near* and *far* bounds are important hyperparameters that can have a big impact on the performance of the method. For the experiments on the DTU dataset, we empirically chose a near bound of 0.85 and a far bound of 1.75 for all scenes and target viewpoints. For the experiments on the RFF, LLFF and IBRNet datasets, we follow the established practice of using the bounds determined by COLMAP [54], with 0.9 and 1.1 factors for the near and far bounds respectively. More generally, the choice of distances $\{a_d\}_{d=1}^{D}$—which determines the distribution of depth planes in the scene—faces similar issues to the choise of sample points along rays in volumetric rendering. While sophisticated sampling strategies exist [3], we chose two standard distributions. We sample the distances uniformly in depth for DTU and ILSH, and uniformly in disparity for RFF, LLFF and IBRNet. For the hyperparameters of the ConvGLR model, we used $D = 128$ and $G = 4$, corresponding to an effective number of depths $D_G = 32$, and $C = 128$ in all our experiments. The model is relatively large with 40M parameters (95M when the parameters of the rendering blocks are not shared), but it is fast, rendering a 375×512 image in 0.71 seconds on a single GPU. Unless stated otherwise, all our models are trained with the Adam optimizer for 120k steps with a learning rate of 1.5e-4, decreased to 1.5e-5 in the last 20% of the training and optionally to 1.5e-6 for the last 5%. We train on patches of 360×360 pixels (or full images for Sparse DTU) with a batch size of 4 or 8 depending on the experiment, using 4 or 8 GPUs respectively. We use a standard VGG loss [14, 41, 78], which we switch to an L1 loss in the last 10% of the training to avoid gridding artifacts. We use gradient clipping to stabilize the training.

5. Experiments

We evaluate our method under sparse and generalizable novel view synthesis scenarios. We consider 5 different experimental setups, using 3 different validation datasets, as detailed below. In all cases, our convolutional global latent renderer (ConvGLR) significantly outperforms the baselines. Qualitative comparisons are available in Fig. 1 and in the Supp. Mat., where an additional evaluation on the Spaces dataset [14] is also presented.

Table 2: Sparse DTU We reproduce the setup introduced in PixelNeRF [76], refined in RegNeRF [44] and used in [11, 18, 55, 56] on the DTU dataset [27]. In this setup, the images are downsampled 4× to a resolution of 400×300. Images with incorrect exposure are excluded.[3] The dataset is split into 88 scenes for training with 7 lighting conditions and 15 scenes for validation.[4] Three scenarios are considered with 3, 6 and 9 input views.[5] Validation is performed on all the views that are not input views or excluded views for all the validation scenes, with lighting condition nb. 3. We report PSNR, SSIM and LPIPS (VGG variant) metrics computed on full and masked images, using object masks produced by [44]. We train 3 different models with 3, 6 and 9 input views on the 88 training scenes (ConvGLR). We see that they outperform all the baseline in all 3 scenarios by significant margins, especially on the full images due to a strong ability to generalize the background across scenes. We then finetune each model once

[2] However, we observe that the PSV is already view-dependent and the angular distance could be computed implicitly by measuring the magnitude of the translations between successive depths.

[3] Images [3, 4, 5, 6, 7, 16, 17, 18, 19, 20, 21, 36, 37, 38, 39].

[4] Scans [8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, 114].

[5] First 3, 6 and 9 images in [25, 22, 28, 40, 44, 48, 0, 8, 13].

Method	Setting	PSNR↑						SSIM↑						LPIPS↓					
		3-view		6-view		9-view		3-view		6-view		9-view		3-view		6-view		9-view	
		full	masked	full	masked	full	masked	full	masked	full	masked	full	masked	full	masked	full	masked	full	masked
SRF [8]	generalizable unknown scene	15.84	15.32	17.77	17.54	18.56	18.35	0.532	0.671	0.616	0.730	0.652	0.752	0.482	0.304	0.401	0.250	0.359	0.232
PixelNeRF [76]		18.74	16.82	21.02	19.11	22.23	20.40	0.618	0.695	0.684	0.745	0.714	0.768	0.401	0.270	0.340	0.232	0.323	0.220
MVSNeRF [5]		16.33	18.63	18.26	20.70	20.32	22.40	0.602	0.769	0.695	0.823	0.735	0.853	0.385	0.197	0.321	0.156	0.280	0.135
ConvGLR (Ours)		20.47	21.57	25.23	23.76	26.98	25.44	0.784	0.846	0.843	0.878	0.878	0.907	0.249	0.159	0.189	0.123	0.147	0.090
SRF ft [8]	generalizable known scene	16.06	15.68	18.69	18.87	19.97	20.75	0.550	0.698	0.657	0.757	0.678	0.785	0.431	0.281	0.353	0.225	0.325	0.205
PixelNeRF ft [76]		17.38	18.95	21.52	20.56	21.67	21.83	0.548	0.710	0.670	0.753	0.680	0.781	0.456	0.269	0.351	0.223	0.338	0.203
MVSNeRF ft [5]		16.26	18.54	18.22	20.49	20.32	22.22	0.601	0.769	0.694	0.822	0.736	0.853	0.384	0.197	0.319	0.155	0.278	0.135
ConvGLR ft (Ours)		20.52	21.80	25.48	24.13	27.31	25.85	0.790	0.853	0.852	0.886	0.883	0.911	0.237	0.147	0.175	0.110	0.139	0.084
mip-NeRF [2]	scene-specific	7.64	8.68	14.33	16.54	20.71	23.58	0.227	0.571	0.568	0.741	0.799	0.879	0.655	0.353	0.394	0.198	0.209	0.092
DietNeRF [24]		10.01	11.85	18.70	20.63	22.16	23.83	0.354	0.633	0.668	0.778	0.740	0.823	0.574	0.314	0.336	0.201	0.277	0.173
RegNeRF [44]		15.33	18.89	19.10	22.20	22.30	24.93	0.621	0.745	0.757	0.841	0.823	0.884	0.341	0.190	0.233	0.117	0.184	0.089
MixNeRF [56]			18.95		22.30		25.03		0.744		0.835		0.879						
FlipNeRF [55]			19.55		22.45		25.12		0.767		0.839		0.882						
DSNeRF [11]		16.90		20.60		22.30		0.570		0.750		0.810							
SparseNeRF [18]			19.55						0.769						0.201				

Table 2. **Sparse DTU.** Scenarios with 3, 6 and 9 input views. We reproduce the values reported by [44] for [2, 5, 8, 24, 44, 76] and the values reported by each for [11, 18, 55, 56]. We do not reproduce the LPIPS values of [11, 55, 56] as they were computed using the AlexNet variant of LPIPS. We also note that the values reported by [11] were computed on the full images. When a value is not available in the original publication, we simply gray the cell out. For each metric, 1st, 2nd and 3rd best-performing methods are highlighted in red, orange and yellow respectively.

on the input views of the 15 validation scenes for 10k steps (ConvVSR ft). To prevent the model from learning an identity function, we continue exposing it to training scenes, where the target views are distinct from the input views. These models further improve their performances on novel views of the validation scenes.

Method	Setting	PSNR↑	SSIM↑	LPIPS↓
SRF [8]	generalizable unknown scene	12.34	0.250	0.591
PixelNeRF [76]		7.93	0.272	0.682
MVSNeRF [5]		17.25	0.557	0.356
ConvGLR (Ours)		19.95	0.700	0.262
SRF ft [8]	generalizable known scene	17.07	0.436	0.529
PixelNeRF ft [76]		16.17	0.438	0.512
MVSNeRF ft [5]		17.88	0.584	0.327
ConvGLR ft (Ours)		20.53	0.711	0.253
mip-NeRF [2]	scene-specific	14.62	0.351	0.495
DietNeRF [24]		14.94	0.370	0.496
RegNeRF [44]		19.08	0.587	0.336
MixNeRF [56]		19.27	0.629	
FlipNeRF [55]		19.34	0.631	
DSNeRF [11]		18.94	0.582	
SparseNeRF [18]		19.86	0.624	0.328

Table 3. **Sparse RFF.** Scenario with 3 input views. We reproduce the values reported by [44] for [2, 5, 8, 24, 44, 76] and the values reported by each for [11, 18, 55, 56]. We do not reproduce the LPIPS values of [11, 55, 56] as they were computed using the AlexNet variant of LPIPS.

Table 3: Sparse RFF We reproduce the setup introduced in RegNeRF [44] and used in [11, 18, 55, 56] on the Real-Forward Facing dataset (RFF) [42] for 3 input views. In this setup, the images are downsampled 8× to a resolution of 504×378. Every 8th image is used for validation, and the 3 input views are selected evenly from the remaining images. We report PSNR, SSIM and LPIPS (VGG) computed on full images. While it was suggested in [44] that the LLFF dataset [41] is too small for training generalizable methods (36 scenes), we found that finetuning a DTU trained model on LLFF provides good performance (ConvPSV). Again, finetuning our model on the set of 8 validation scenes improves performance futher (ConvGLR ft).

Method	Setting	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF [76]	generalizable unknown scene	19.31	0.789	0.671
IBRNet [70]		26.04	0.917	0.190
MVSNeRF [5]		26.63	0.931	0.168
GPNR [63]		28.50	0.932	0.167
ConvGLR (Ours)		31.65	0.952	0.080

Table 4. **Generalizable DTU.** We reproduce the values reported by [5] for [5, 70, 76] and the value reported by [63].

Table 4: Generalizable DTU We reproduce the setup introduced in MVSNeRF [5] and used in GPNR [63] on the DTU dataset [27]. In this setup, the images are downsampled 2× and cropped to a resolution of 640×512 (images pre-processed by MVSNet [74]). The dataset is split into 88 scenes for training with 7 lighting conditions and 16 scenes for validation[6]. The images with incorrect exposure are not excluded during training. One scenario is considered with 10 input views, using the input/target split from [5]. Validation is performed on 4 views per scene[7] with lighting condition nb. 3. We report PSNR, SSIM and LPIPS (VGG) metrics computed on masked images (foreground pixels, whose ground truth depths stand inside the scene bound). Our model significantly outperforms previous methods.

[6] Scans [1, 8, 21, 30, 31, 34, 38, 40, 41, 45, 55, 63, 82, 103, 110, 114].
[7] images [23, 24, 32, 44].

Method	Setting	PSNR↑	SSIM↑	LPIPS↓
LLFF [41]	generalizable unknown scene	24.13	0.798	0.212
IBRNet [70]		25.13	0.817	0.205
GeoNeRF [28]		25.44	0.839	0.180
GPNR [63]		25.72	0.880	0.175
ConvGLR (Ours)		26.94	0.875	0.164
SRN [59]	generalizable known scene	22.84	0.668	0.378
IBRNet ft [70]		26.73	0.851	0.175
GeoNeRF ft 10k [28]		26.58	0.856	0.162
ConvGLR ft (Ours)		27.81	0.889	0.125
NeRF [42]	scene-specific	26.50	0.811	0.250
GRF [67]		26.64	0.837	0.178

Table 5. **Generalizable RFF.** We reproduce the values reported by [42] for [41, 42, 59] and the values reported by each for [28, 63, 67, 70].

Table 5: Generalizable RFF We reproduce the setup introduced in NeRF [42] and used in [28, 63, 67, 70] on the Real Forward-Facing (RFF) dataset [42]. In this setup, the images are downsampled 4× to a resolution of 1008×756. Every 8th image is used for validation, and 10 nearby input views are selected from the remaining images. We report PSNR, SSIM and LPIPS (VGG) computed on full images. We finetune our DTU-trained model for 50k steps on the IBRNet dataset (ConvGLR). We then finetune the model for another 4k steps on the 8 validation scenes (ConvGLR ft).

Method	PSNR↑		SSIM↑		Time↓ (s)
	full	masked	full	masked	
C0:TensoRF	20.54	26.17	0.71	0.82	94.02
T3:CogCoVi	21.49	26.33	0.70	0.82	806.00
T2:NoNeRF	20.37	26.43	0.69	0.82	175.58
T1:OpenSpaceAI	21.66	27.02	0.68	0.83	76.88
C2:DINER-SR	22.37	28.50	0.72	0.83	87.25
C1:MPFER-H	28.05	28.90	0.84	0.83	1.50
ConvGLR (Ours)	28.39	30.17	0.85	0.84	0.71

Table 6. **ILSH dataset.** We reproduce the values from the ICCV 2023 view synthesis challenge: *To NeRF or not to NeRF* [26].

Table 6: ILSH The Imperial Light-Stage Head dataset (ILSH) [77] was introduced as a benchmark for a recent ICCV 2023 view synthesis challenge [26]. The dataset consists in 52 scenes (one individual per scene) with 24 views each at a resolution of 3000×4096, with 50 views from 38 scenes held out for testing. The dataset is publicly available upon request and blind evaluation on the test set can be performed on the Codalab platform [25]. Evaluation is performed using PSNR and SSIM metrics, on full and masked images. Following the challenge organising team C1:MPFER-H [26], we downsample the images 8× and train our model on the 52 scenes using 16 input views. Our method outperforms the challenge winner T1:OpenSpaceAI and the challenge organizing team C1:MPFER-H by more than 3dB and 1.2dB in masked PSNR respectively (metric used during the challenge).

line	pos. enc.	ang. enc.	backbone	patch size	params	FLOPS	PSNR↑
1	**Yes**	**Yes**	No PSV	**256×256**	29.6M	0.3T	17.30
2	**Yes**	**Yes**	MVS-based	**256×256**	40.1M	6.1T	23.39
3	**Yes**	**Yes**	MPI-based	**256×256**	28.2M	7.8T	24.62
4	**Yes**	**Yes**	**ConvGLR**	16×16	40.3M	6.6T	24.03
5	**Yes**	**Yes**	**ConvGLR**	32×32	40.3M	6.6T	25.79
6	**Yes**	**Yes**	**ConvGLR**	64×64	40.3M	6.6T	26.22
7	**Yes**	**Yes**	**ConvGLR**	128×128	40.3M	6.6T	26.20
8	No	No	**ConvGLR**	**256×256**	40.2M	6.5T	25.66
9	**Yes**	No	**ConvGLR**	**256×256**	40.3M	6.5T	25.91
10	**Yes**	**Yes**	**ConvGLR**	**256×256**	40.3M	6.6T	26.33

Table 7. **Ablations.** All the models were trained on the Sparse DTU setup with 9 input views for 50k steps.

Table 7: Ablations We perform ablations on the Sparse DTU setup with 9 input views, and train each model for 50k steps on patches of 256×256 pixels. We start by training our full model (line 10). We then consider 3 variants of our backbone architecture. *No PSV*: the input images are concatenated and processed as a group, but no PSV is constructed (line 1). *MVS-based*: deep features are extracted from individual input images and a cost volume is constructed by computing the variance over the views (line 2). *MPI-based*: the model outputs D RGBα images that are then alpha-blended (line 3). We see that our ConvGLR backbone produces the best results by big margins, validating our choice of a PSV based architecture rendering novel views globally in a low-dimensional latent space. We then train the same model 4 times, on image patches ranging from 16×16 to 128×128 (lines 4-7). In order to keep the effective batch size constant, we train on 256×256 patches that we slice into 16^2, 8^2, 4^2 and 2^2 pieces respectively. We see that the performance degrades sharply with smaller patch sizes, confirming that the global rendering contributes significantly to the performance of our approach. Finally we turn off the positional and angular encodings together and separately (lines 8-9). We see that both contribute to the final performance of the model.

6. Conclusion

We introduced global latent neural rendering, a novel view synthesis approach that consists in learning a generalizable light field model from plane sweep volumes, and ConvGLR, a convolutional architecture that implements this idea efficiently. While ConvGLR performs remarkably well, we believe that there is still room for improvement by optimizing the architecture, scaling up the training, and sampling the depth planes in a scene-adaptive manner. Another interesting direction for future work is the application to very high-resolution images, where training on large image portions becomes challenging due to memory constraints during training.

References

[1] Benjamin Attal, Jia-Bin Huang, Michael Zollhöfer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding. In *CVPR*, pages 19819–19829, 2022. 2, 4

[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 *ICCV*, pages 5855–5864, 2021. 1, 2, 3, 4, 7

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 1, 2, 3, 4, 6

[4] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *CVPR*, pages 4947–4956, 2021. 6

[5] 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. 1, 2, 3, 4, 5, 7

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, pages 333–350. Springer, 2022. 1, 2, 3, 4

[7] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *CVPR*, pages 2524–2534, 2020. 3

[8] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *CVPR*, pages 7911–7920, 2021. 3, 7

[9] Robert T Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363, 1996. 3, 4, 12

[10] P.E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *ACM Transactions on Graphics (TOG)*, pages 11–20, 1996. 3

[11] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, pages 12882–12891, 2022. 1, 2, 3, 6, 7

[12] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *CVPR*, 2023. 2, 3, 4, 5

[13] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, pages 5515–5524, 2016. 3, 4, 5

[14] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, pages 2367–2376, 2019. 2, 3, 4, 5, 6, 12, 13

[15] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022. 1, 2

[16] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, page 43–54, 1996. 2

[17] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, pages 2495–2504, 2020. 3

[18] Guangcong, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *ICCV*, 2023. 1, 2, 3, 6, 7

[19] Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-view view synthesis in the wild with learned adaptive multi-plane images. In *ACM SIGGRAPH*, 2022. 3

[20] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 4

[21] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3d photography. *ACM Transactions on Graphics (TOG)*, 36(6):1–15, 2017. 3

[22] Ronghang Hu, Nikhila Ravi, Alexander C Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *ICCV*, pages 12528–12537, 2021. 3

[23] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In-So Kweon. Dpsnet: End-to-end deep plane sweep stereo. In *ICLR*, 2019. 3

[24] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, pages 5885–5894, 2021. 2, 7

[25] Youngkyoon Jang, Jiali Zheng, Jiankang Deng, Ales Leonardis, and Stefanos Zafeiriou. To nerf or not to nerf. https://codalab.lisn.upsaclay.fr/competitions/14427, 2023. 2, 8

[26] Youngkyoon Jang, Jiali Zheng, Jifei Song, Helisa Dhamo, Eduardo Pérez-Pellitero, Thomas Tanay, Matteo Maggioni, Richard Shaw, Sibi Catley-Chandar, Yiren Zhou, et al. Vschh 2023: A benchmark for the view synthesis challenge of human heads. In *ICCV*, pages 1121–1128, 2023. 1, 2, 8

[27] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, pages 406–413, 2014. 2, 6, 7

[28] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *CVPR*, pages 18365–18375, 2022. 1, 2, 3, 4, 5, 8, 15

[29] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM TOG*, 35(6):1–10, 2016. 2, 3, 5

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 2

[31] Taras Khakhulin, Denis Korzhenkov, Pavel Solovev, Gleb Sterkin, Andrei-Timotei Ardelean, and Victor Lempitsky. Stereo magnification with multi-layer images. In *CVPR*, pages 8687–8696, 2022. 3

[32] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, page 31–42, 1996. 2

[33] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *ICCV*, pages 12578–12588, 2021. 3

[34] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, 2023. 2, 5

[35] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR workshop*, pages 136–144, 2017. 6

[36] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *ACM TOG*, pages 1–9, 2022. 3, 4, 5

[37] Kai-En Lin, Zexiang Xu, Ben Mildenhall, Pratul P Srinivasan, Yannick Hold-Geoffroy, Stephen DiVerdi, Qi Sun, Kalyan Sunkavalli, and Ravi Ramamoorthi. Deep multi depth panoramas for view synthesis. In *ECCV*, pages 328–344. Springer, 2020. 3

[38] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. 31, 2018. 6

[39] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Theobalt Christian, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022. 3, 4, 5

[40] Leonard McMillan Jr. *An image-based approach to three-dimensional computer graphics*. The University of North Carolina at Chapel Hill, 1997. 3

[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 TOG*, 38 (4):1–14, 2019. 1, 3, 4, 5, 6, 7, 8

[42] 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 *ECCV*, 2020. 1, 2, 3, 4, 6, 7, 8

[43] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 41(4):1–15, 2022. 1, 2, 3, 4

[44] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, pages 5480–5490, 2022. 1, 2, 6, 7, 14

[45] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. 36(6), 2017. 3, 5

[46] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM TOG*, 36(6):1–11, 2017. 12, 13

[47] Malte Prinzler, Otmar Hilliges, and Justus Thies. Diner: Depth-aware image-based neural radiance fields. In *CVPR*, pages 12449–12459, 2023. 3

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. pages 8748–8763, 2021. 2

[49] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, pages 10901–10911, 2021. 2

[50] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, pages 623–640. Springer, 2020. 3

[51] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, pages 12216–12225, 2021. 3

[52] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Ganerf: Leveraging discriminators to optimize neural radiance fields. *ACM Trans. Graph.*, 2023. 2

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[54] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016. 2, 6

[55] Seunghyeon Seo, Yeonjin Chang, and Nojun Kwak. Flipnerf: Flipped reflection rays for few-shot novel view synthesis. In *ICCV*, pages 22883–22893, 2023. 2, 6, 7

[56] Seunghyeon Seo, Donghoon Han, Yeonjin Chang, and Nojun Kwak. Mixnerf: Modeling a ray with mixture density for novel view synthesis from sparse inputs. In *CVPR*, pages 20659–20668, 2023. 2, 6, 7

[57] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998. 3

[58] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, pages 2–13. SPIE, 2000. 2, 3

[59] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. 32, 2019. 8

[60] Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. 34: 19313–19325, 2021. 2, 4

[61] Pavel Solovev, Taras Khakhulin, and Denis Korzhenkov. Self-improving multiplane-to-layer images for novel view synthesis. In *WACV*, pages 4309–4318, 2023. 3

[62] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the

boundaries of view extrapolation with multiplane images. In *CVPR*, pages 175–184, 2019. 3, 4

[63] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*, 2022. 1, 2, 4, 5, 7, 8, 15

[64] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *CVPR*, pages 8269–8279, 2022. 2, 4, 5

[65] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *ICCV*, pages 517–524, 1998. 3, 4

[66] Thomas Tanay, Ales Leonardis, and Matteo Maggioni. Efficient view synthesis and 3d-based multi-frame denoising with multiplane feature representations. In *CVPR*, 2023. 3, 12, 13

[67] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *CVPR*, pages 15182–15192, 2021. 2, 4, 8

[68] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, pages 551–560, 2020. 3

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017. 6

[70] 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. 1, 2, 4, 5, 7, 8

[71] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, pages 8534–8543, 2021. 3

[72] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *AAAI*, pages 12508–12515, 2020. 3

[73] Jiayu Yang, Wei Mao, Jose M Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, pages 4877–4886, 2020. 3

[74] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, pages 767–783, 2018. 3, 7

[75] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *CVPR*, pages 5525–5534, 2019. 3

[76] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021. 1, 2, 4, 6, 7

[77] Jiali Zheng, Youngkyoon Jang, Athanasios Papaioannou, Christos Kampouris, Rolandos Alexandros Potamias, Foivos Paraperas Papantoniou, Efstathios Galanakis, Aleš Leonardis, and Stefanos Zafeiriou. Ilsh: The imperial light-stage head dataset for human head view synthesis. In *ICCV*, pages 1112–1120, 2023. 2, 8

[78] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG*, 37(4):1–12, 2018. 3, 4, 5, 6

Global Latent Neural Rendering

Supplementary Material

7. Averaged plane sweep volume

As discussed in Sec. 4, the plane sweep volume is a highly structured tensor encoding the epipolar geometry between the input views and the target view. One of its interesting properties is that local image features match across the input views when a depth plane is precisely located on an object in the scene. A simple way to highlight this property is to average the PSV over the input views, as done in Fig. 4. There, each depth plane slices the 3D object at a specific depth. When a part of the object is located on the depth plane, this part appears "in focus" in the mean PSV. On the contrary, the parts that are located at other depths appear blurry and out of focus. Such averaging of the PSV is closely related to the original plane sweep algorithm of Collins [9] for depth estimation, and further motivates the use of plane sweep volumes for novel view synthesis.

8. Implementation details

We presented an overview of our Convolutional Global Latent Renderer (ConvGLR) in Sec. 4 and Fig. 3 of the main paper. ConvGLR transforms 5D input PSVs into 3D rendered images in 4 steps: (1) Grouped PSV, (2) Multi-view matching, (3) global latent rendering and (4) upsampling. We provide more details in Tab. 8 where all the operations are listed with their effect on the dimension of the input tensor. Particular emphasis has been put on memory efficiency and in-place viewing operations are used extensively while expensive reshape or transpose operations are avoided.

We propose two possible implementations of the global latent rendering step: one where the resblocks are applied over the depths with shared weights by using the batch dimension for parallel processing, and one where the resblocks are applied over the depths with specialized weights by moving the depths into the channel dimension and applying resblocks implemented with grouped convolutions. In practice, we did not observe any significant difference of performance between the two implementations.

9. Spaces dataset

Table 9: Spaces. We reproduce the setup from DeepView [14] and used in MPFER [66] on the Spaces dataset [14]. This dataset consists of 100 indoor and outdoor scenes, captured 5 to 10 times each using a 16-camera rig translated by small amounts. The dataset is split into 90 scenes for training and 10 scenes for validation. The resolution of the images is 480×800. Four scenarios are considered: one with 12 input views and three with 4 input views. Following MPFER [66], we train one model for the scenario with 12 input views and one model for the 3 scenarios with 4 input views. Validation is performed on the first rig position for the 10 validation scenes, on the target images specified in [14] for each scenario. We report PSNR, SSIM and LPIPS (AlexNet variant) metrics computed on images after cropping an outer boundary of 16 pixels as done in [14, 66]. Our Convolutional Global Latent Renderer (ConvGLR) outperforms Soft3D [46], DeepView [14] and MPFER [66] by significant margins in all scenarios.

10. Qualitative results

We provide a number of qualitative comparisons to baselines in Fig. 5, Fig. 6, Fig. 7, Fig. 8.

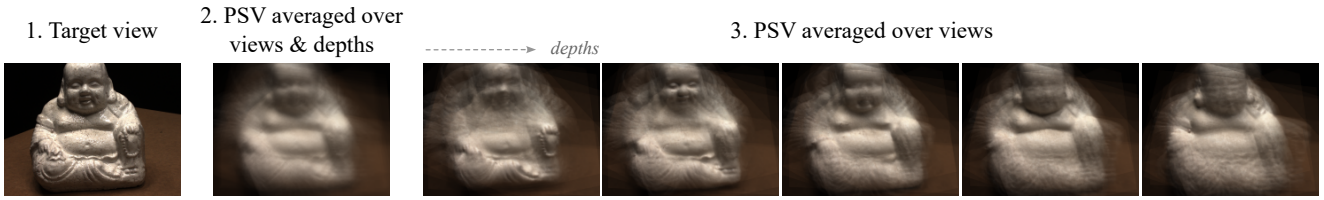

Figure 4. **Averaging the plane sweep volume.** 1. The target view for which a plane sweep volume is constructed, using 9 input views (not including the target view) and *near* and *far* bounds that are close to the object depth. 2. Averaging the PSV over views and depths provides a blurry estimate of the target views. 3. Averaging the PSV over the views brings successive depths of the object in focus.

implementation	shared weights					specialized weights				
block	description	output dimension				description	output dimension			
		batch	channels	height	width		batch	channels	height	width
grouped PSV	5D PSV	$D\,V$	3	H	W					
	concatenate views	D	$3V$	H	W					
	view	D_G	$3GV$	H	W					
multi-view matching	**conv.**	D_G	C	H	W					
	2 resblocks	D_G	C	H	W					
	conv. (stride 2)	D_G	$2C$	$H/2$	$W/2$					
	3 resblocks	D_G	$2C$	$H/2$	$W/2$					
	conv. (stride 2)	D_G	$4C$	$H/4$	$W/4$					
	4 resblocks	D_G	$4C$	$H/4$	$W/4$					
global latent rendering	view	$D_G/2$	$8C$	$H/4$	$W/4$	view	1	$D_G\times 4C$	$H/4$	$W/4$
	1 resblock	$D_G/2$	$4C$	$H/4$	$W/4$	**1 resblock** ($D_G/2$ groups)	1	$D_G/2\times 4C$	$H/4$	$W/4$
	view	$D_G/4$	$8C$	$H/4$	$W/4$					
	1 resblock	$D_G/4$	$4C$	$H/4$	$W/4$	**1 resblock** ($D_G/4$ groups)	1	$D_G/4\times 4C$	$H/4$	$W/4$
	view	$D_G/8$	$8C$	$H/4$	$W/4$					
	1 resblock	$D_G/8$	$4C$	$H/4$	$W/4$	**1 resblock** ($D_G/8$ groups)	1	$D_G/8\times 4C$	$H/4$	$W/4$
	view	$D_G/16$	$8C$	$H/4$	$W/4$					
	1 resblock	$D_G/16$	$4C$	$H/4$	$W/4$	**1 resblock** ($D_G/16$ groups)	1	$D_G/16\times 4C$	$H/4$	$W/4$
	view	1	$D_G/16\times 4C$	$H/4$	$W/4$					
	1 resblock	1	$4C$	$H/4$	$W/4$	**1 resblock**	1	$4C$	$H/4$	$W/4$
upsampling	interpolate (nearest)	1	$4C$	$H/2$	$W/2$					
	3 resblocks	1	$2C$	$H/2$	$W/2$					
	interpolate (nearest)	1	$2C$	H	W					
	2 resblocks	1	C	H	W					
	conv.	1	3	H	W					

Table 8. **ConvGLR.** The 5D plane sweep volume is progressively turned into a 3D rendered image by applying a succession of 2D convolutions and resblocks while making effective use of viewing operations and batching. Learnable blocks are emphasized in bold.

Method	PSNR↑				SSIM↑				LPIPS↓			
	12 views	4 views			12 views	4 views			12 views	4 views		
	dense	small	medium	large	dense	small	medium	large	dense	small	medium	large
Soft3D [46]	31.93	30.29	30.84	30.57	0.940	0.925	0.930	0.931	0.052	0.064	0.060	0.054
DeepView [14]	34.23	31.42	32.38	31.00	0.965	0.954	0.957	0.952	0.015	0.026	0.021	0.024
MPFER [66]	35.73	33.20	33.47	32.38	0.972	0.959	0.959	0.953	0.012	0.018	0.018	0.021
ConvGLR (Ours)	36.05	34.07	34.33	33.34	0.977	0.968	0.968	0.964	0.013	0.018	0.018	0.020

Table 9. **Spaces.** We reproduce the values reported by [66] for [14, 46, 66] (computed on images provided by the authors for [14, 46]). LPIPS values were computed with the AlexNet backbone following [66].

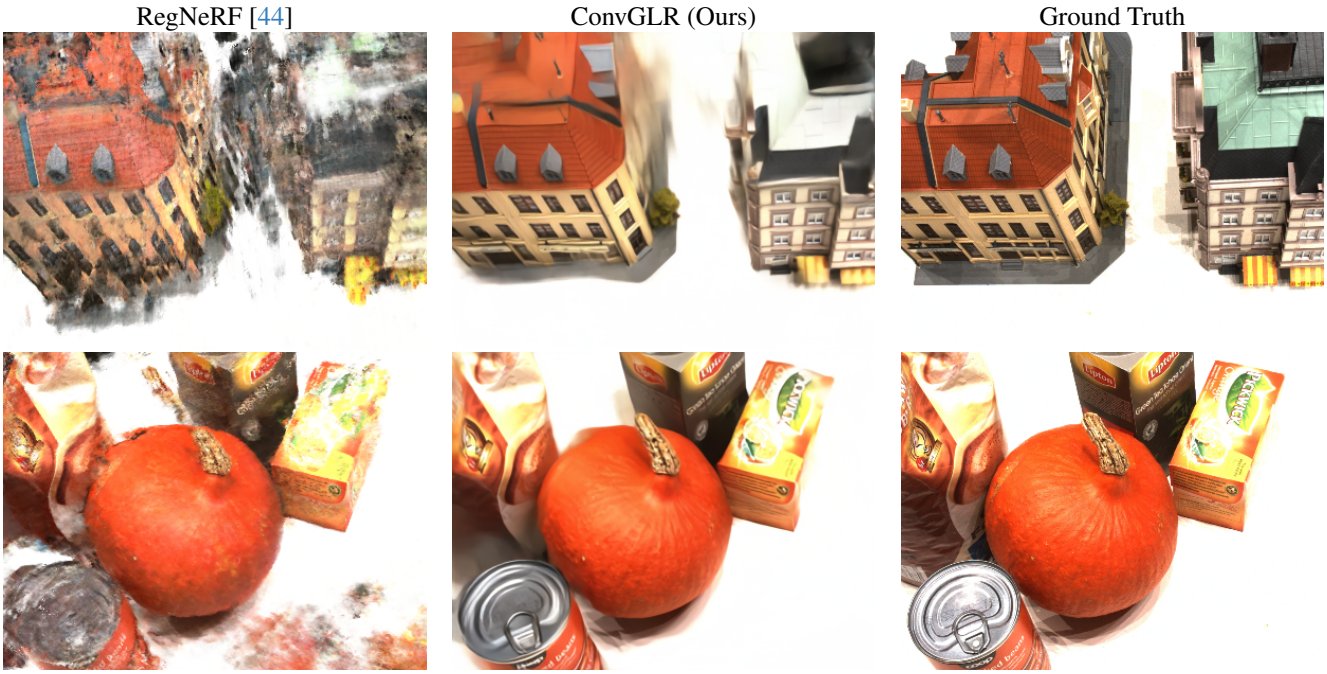

Figure 5. **Qualitative results.** Sparse DTU.

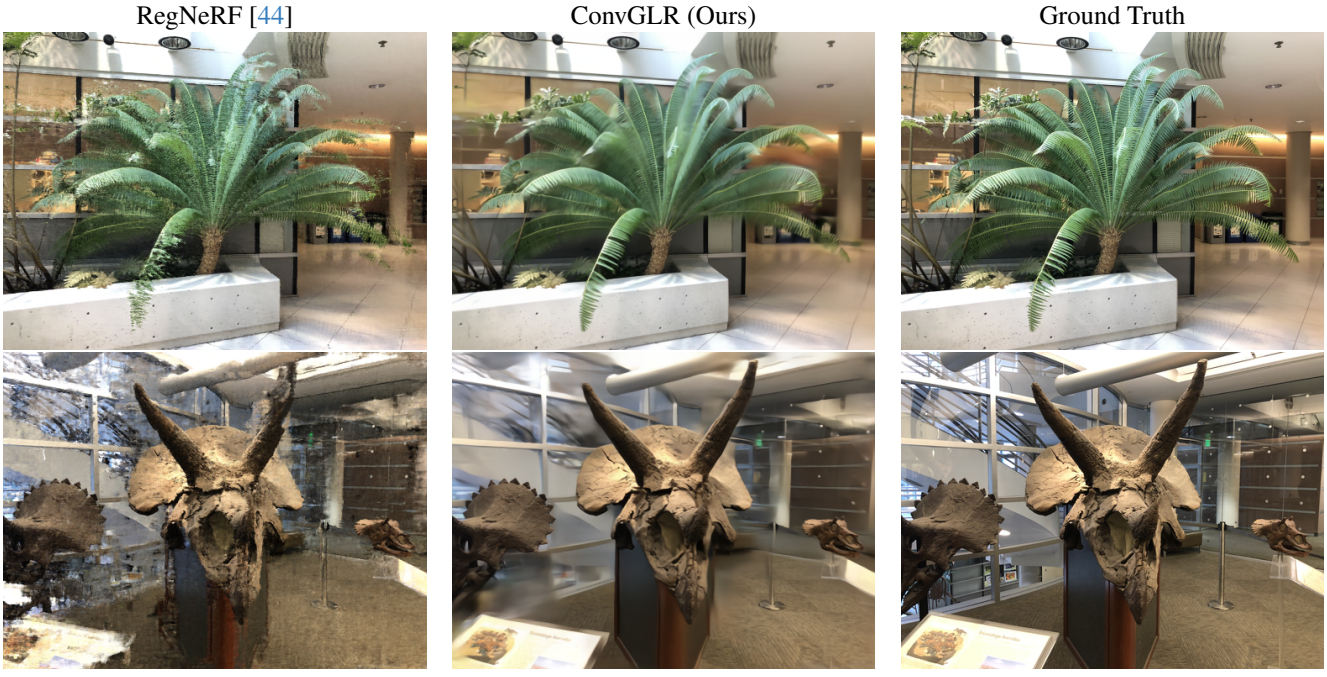

Figure 6. **Qualitative results.** Sparse RFF.

Figure 7. **Qualitative results.** Generalizable DTU (unknown scenes).

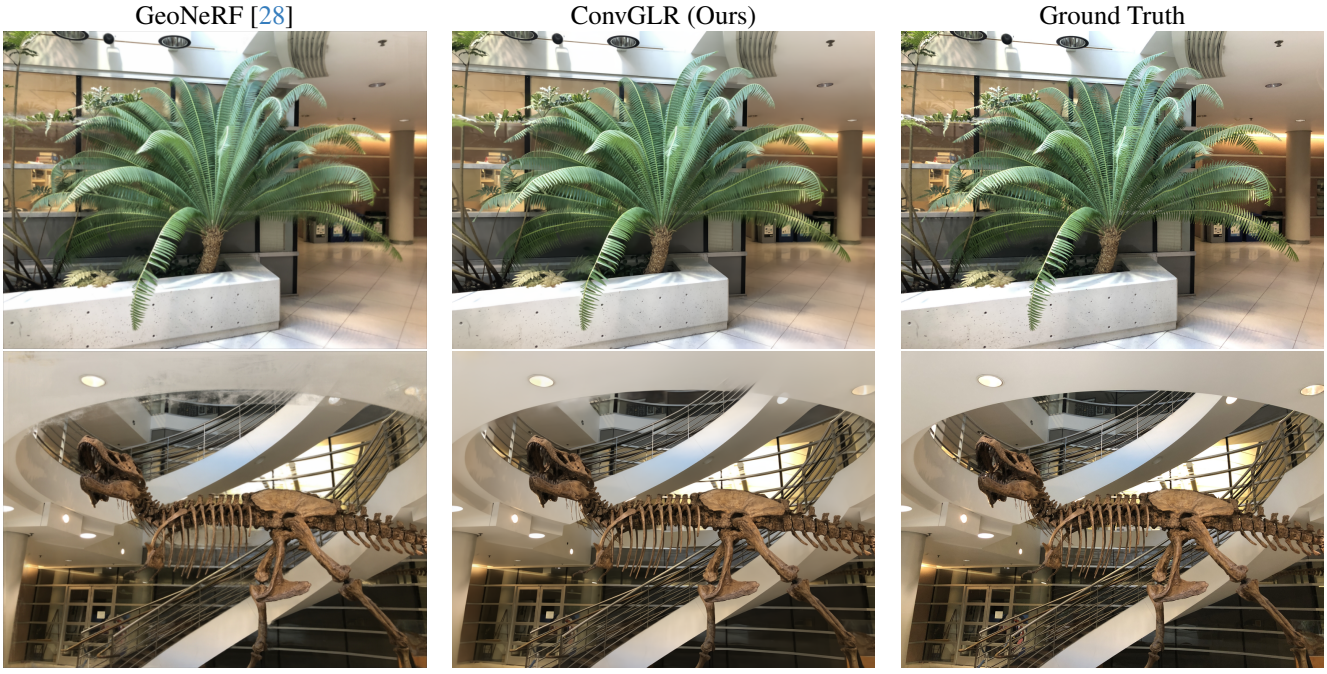

Figure 8. **Qualitative results.** Generalizable RFF (unknown scenes).