Self-improving Multiplane-to-layer Images for Novel View Synthesis

Pavel Solovev[1*] Taras Khakhulin[1,2*] Denis Korzhenkov[3*]

[1]Samsung AI Center – Moscow
[2]Skolkovo Institute of Science and Technology
[3] Yandex Research

https://samsunglabs.github.io/MLI/

Abstract

We present a new method for lightweight novel-view synthesis that generalizes to an arbitrary forward-facing scene. Recent approaches are computationally expensive, require per-scene optimization, or produce a memory-expensive representation. We start by representing the scene with a set of fronto-parallel semitransparent planes and afterwards convert them to deformable layers in an end-to-end manner. Additionally, we employ a feed-forward refinement procedure that corrects the estimated representation by aggregating information from input views. Our method does not require any fine-tuning when a new scene is processed and can handle an arbitrary number of views without any restrictions. Experimental results show that our approach surpasses recent models in terms of both common metrics and human evaluation, with the noticeable advantage in inference speed and compactness of the inferred layered geometry.

1. Introduction

A problem of novel view synthesis (NVS) consists in predicting the view I_n of a scene from a novel camera viewpoint π_n, given a set of input views $\{I_i\}_{i=1}^V$ for that scene (also referred to as source views) and the corresponding camera poses $\{\pi_i\}_{i=1}^V$ and intrinsic parameters $\{K_i\}_{i=1}^V$ [10]. Two natural origins for such input views are handheld videos of static scenes [25, 44, 21] and shots from a multi-camera rig [7, 2]. To this day, the best image quality is obtained by estimating the radiance field of the scene, deriving it from frames of the source video that are the closest to the novel camera pose [25, 47, 1, 27]. Nevertheless, such approaches require fine-tuning of the model on a new scene to achieve the best results. This limitation prevents them from being used in settings where fast rendering is needed.

On the other hand, a class of methods based on *multiplane images* (MPI) [50, 40, 35] provides real-time rendering and good generalization, by representing the scene with a set of fronto-parallel planes given several input images [24, 7]. One of their restrictions is the relatively large number of semitransparent planes required to approximate the scene geometry. To cope with this, recent works [2, 20] proposed to generate a dense set of planes (as many as 128) and merge them into a *multilayer image* (MLI) with a non-learnable post-processing operation. This research direction was followed by the methods which estimate the MLI end-to-end with a neural network [12, 15].

In this paper, we present a new method for photorealistic view synthesis, which estimates a multilayer geometry of the scene in the form of an MLI, given an arbitrary set of forward-facing views. A network that predicts a proxy geometry is trained on a dataset of scenes, and after that, MLI is obtained in a feed-forward fashion for any new scene. In contrast with prior solutions, our method is free of a predefined number of input views or any neural computation during rendering. Thus, the output representation can be rendered in high resolution with standard graphics engines even on devices with computational restrictions.

We refer to the proposed model as **S**elf-**i**mproving **M**ulti**p**lane-to-**l**ayer **I**mage, or just SIMPLI. The basic steps of our approach are outlined in the name: First, we estimate the geometry of a scene in the form of the multiplane image, which is converted into the multilayer image in an *end-to-end* manner, see Fig. 2. Additionally, we refine the representation with a feed-forward *error correction* procedure, inspired by the DeepView paper [7], based on input views (hence *self-improving*). Our evaluation shows that for test scenes not seen during training, we can synthesize images on par with or superior to state-of-the-art methods designed to generalize to new scenes. Moreover, our method provides faster inference speed.

*All authors contributed equally.

Figure 1. The MLI representation estimated by SIMPLI with 4 deformable layers. Semitransparent layers are enumerated in the back-to-front order. The inferred depth map is computed by overcomposing the per-layer depth maps w.r.t. the opacity extracted from the corresponding RGBA textures.

2. Related works

Novel view synthesis for a static scene is a well-known problem of computer vision. The main task in this field is to produce plausible images corresponding to the camera's motion through the scene, based on the input information. Early methods interpolated directly between the pixels of the input images with restrictions on the positions of the source cameras [8, 14, 11], used the proxy geometry of the scene [5] or optimization accompanied with heuristics [28] to render the scene from a new viewpoint.

The approach called *Stereo magnification* used MPI geometry with semitransparent planes placed in the frustum of one of the source cameras [50]. The *DeepView* method refined the textures of planes step-by-step with a procedure similar to the learned gradient descent [7, 2]. Although it was reportedly able to handle any number of input images, this was not demonstrated. The authors of the *LLFF* method [24] built a separate MPI in the frustum of each source camera and used a heuristic to blend multiple preliminary novel views obtained using those MPIs to get the resulting image. In contrast, we aggregate information from an arbitrary number of input views and construct a single representation for a scene.

Several works [33, 6] found the usage of layered depth image (LDI) [32] efficient for single-image NVS, due to its sparsity and simplicity of rendering. At the same time, estimating this representation in a differentiable way without relaxation is still challenging [41]. *StereoLayers* [15] and *Worldsheet* [12] methods represented a scene with a number of semitransparent deformable layers, a structure also called the "multilayer image", which is less compact than LDI (see the discussion on terminology in Appendix A). Nevertheless, due to its scene-adaptive nature, this proxy geometry is still much more lightweight than MPI while preserving the ability to render in real time with modern graphic engines. Some prior works proposed to convert MPI to MLI with a postprocessing procedure, based on a heuristic [2, 20]. Unlike them, we perform this conversion end-to-end, using the techniques applied in the StereoLayeres [15] paper to the case of only two input frames.

Recently, several methods [48, 3, 39, 44, 4, 37] have attempted to estimate the implicit 3D representation from input images in the form of a neural radiance field [25]. However, massive queries to the neural network at inference time result in slow rendering speed. Moreover, *IBRNet* [44] additionally applied the self-attention operation along the ray to estimate the volumetric density of the points, further slowing the speed. Another group of methods [46, 31, 36, 26] improved the speed of querying and training convergence for a single scene, but these approaches have not been generalized to new scenes. In contrast, our system allows rendering novel views at higher speed while generalizing across various scenes.

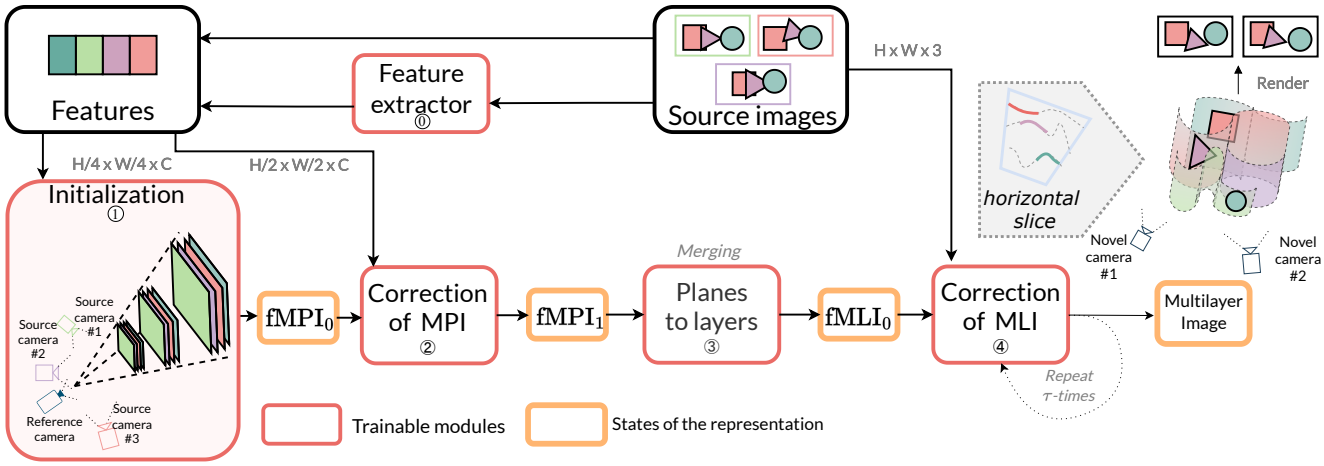

Figure 2. Scheme of the proposed SIMPLI model. Our multistage system consists of 4 steps: ① a preliminary multiplane representation (MPI) is initialized and ② refined with an error correction step. Afterwards, ③ MPI is converted to the multilayer geometry (MLI), which is again passed through an error correction procedure ④. The whole pipeline is implemented in the coarse-to-fine manner.

3. Preliminaries

MPI representation. MPI represents a scene with a sequence of P planes, placed in the frustum of a pinhole camera, referred to as a *reference* camera. We construct a virtual reference camera by averaging the poses of the source cameras [23]. These P planes are placed with uniform disparity in the predefined depth range, which depends on the preprocessing of the dataset. In our experiments $P = 40$.

Each plane has a semitransparent RGBA texture of resolution $h \times w$, disregarding its depth. To render the final image using this representation, the planes' textures are "projected" to the novel camera pose using homography warping and composed-over w.r.t. their opacities [29]. Therefore, rendering an arbitrary number of new images is fast and does not involve any neural networks. The critical question here is how to estimate the textures of the planes. Typically, this is done by processing a *plane-sweep volume* (PSV).

Building the PSV. Here we define the procedure of building a plane-sweep volume $PSV = \texttt{unproj}\left(\{F_i\}_{i=1}^V, h, w\right)$, given a set of V source feature tensors $\{F_i\}_{i=1}^V$ consisting of C channels and the resolution of MPI planes $h \times w$. The features $\{F_i\}$ may coincide with the source images $\{I_i\}$ or be the outputs of some encoding network. To construct the volume, each of the feature tensors is "unprojected" onto the planes with homography warping. The result of the procedure $\texttt{unproj}$ is a tensor of shape $V \times P \times C \times h \times w$. Informally speaking, to obtain MPI from the built PSV, we need to "reduce" the V axis, *i.e.* to aggregate all source features, and subsequently convert them to RGBA domain [50, 7].

Attention pooling. Attention pooling [19] is a modification of a standard QKV-attention module [43], where queries are represented with trainable vectors called *anchors*, independent of input, while keys and values are equal to input vectors. This operation allows to compress an arbitrary number of inputs to the predefined number of outputs.

4. Method

4.1. Overview

① **Initialization.** First, we process each of the V source views with a feature extractor based on the feature pyramid architecture $E_\theta : I_i \mapsto F_i'$ yielding V tensors of the same resolution $H \times W$ as original images. Each tensor F_i' is concatenated with I_i channel-wise, providing a feature tensor F_i. The features $\{F_i\}_i$ are used to build the plane-sweep volume PSV_0 of resolution $\frac{H}{4} \times \frac{W}{4}$ with P planes. The volume serves as input for the aggregation module T_0 (described below). The output of this module is an initial version of MPI in the feature domain, denoted as $fMPI_0$.

② **Correction of MPI.** Then we conduct an error correction step. We project $fMPI_0$ on the source cameras at resolution of $\frac{H}{2} \times \frac{W}{2}$ and compute the difference with feature tensors $\{F_i\}$, downsampled to the same size. Using the "unprojected" tensor of discrepancies of shape $V \times P \times C \times \frac{H}{2} \times \frac{W}{2}$, we update the representation to the state $fMPI_1$.

③ **Planes-to-layer conversion.** At this step, we merge the P rigid planes of $fMPI_1$ into L deformable layers (see the detailed diagram in Fig. S1). The procedure is performed in two steps: First, attention pooling (see Sec. 3) with L anchors is applied along the P axis to aggregate planes into L groups. The resulting tensor contains textures in the feature domain for the layers. Second, the depth maps for the deformable layers are predicted: a self-attention module is applied along the P axis of $fMPI_1$ to predict the pixel-wise opacity for each plane. Then we divide the planes into

# source views	Model	SWORD			Real Forward-Facing			Shiny		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
2	IBRNet	$19.02_{4.59}$	$0.54_{0.19}$	$0.35_{0.19}$	$19.13_{3.07}$	$0.57_{0.14}$	$0.32_{0.13}$	$21.89_{3.76}$	$0.67_{0.15}$	$0.24_{0.14}$
	StereoMag	$18.71_{3.95}$	$0.53_{0.19}$	$0.29_{0.18}$	$17.22_{2.85}$	$0.47_{0.15}$	$0.31_{0.12}$	$20.11_{3.63}$	$0.58_{0.16}$	$0.19_{0.11}$
	DeepView†	$20.41_{4.04}$	$0.64_{0.17}$	$0.22_{0.14}$	$20.46_{3.00}$	$0.65_{0.13}$	$0.20_{0.08}$	$22.96_{3.73}$	$0.72_{0.14}$	$0.12_{0.07}$
	SIMPLI-4L	$20.78_{3.83}$	$0.64_{0.16}$	$0.23_{0.15}$	$20.46_{3.00}$	$0.65_{0.13}$	$0.20_{0.08}$	$22.96_{3.73}$	$0.72_{0.14}$	$0.12_{0.07}$
	SIMPLI-8L	$\mathbf{20.84}_{3.76}$	$\mathbf{0.64}_{0.16}$	$\mathbf{0.22}_{0.14}$	$\mathbf{21.17}_{3.09}$	$\mathbf{0.69}_{0.12}$	$\mathbf{0.16}_{0.06}$	$\mathbf{23.59}_{3.27}$	$\mathbf{0.76}_{0.12}$	$\mathbf{0.10}_{0.05}$
5	IBRNet	$22.79_{3.92}$	$0.71_{0.15}$	$0.22_{0.12}$	$22.69_{3.35}$	$0.73_{0.10}$	$0.19_{0.08}$	$25.29_{3.30}$	$0.80_{0.09}$	$0.13_{0.07}$
	LLFF	$19.56_{3.19}$	$0.52_{0.17}$	$0.33_{0.11}$	$21.76_{3.02}$	$0.72_{0.10}$	$0.20_{0.07}$	$23.31_{2.74}$	$0.75_{0.11}$	$0.16_{0.05}$
	DeepView†	$21.99_{3.85}$	$0.73_{0.14}$	$0.18_{0.12}$	$23.11_{2.86}$	$0.76_{0.10}$	$0.13_{0.05}$	$24.99_{3.24}$	$0.81_{0.09}$	$0.09_{0.04}$
	SIMPLI-4L	$22.95_{3.01}$	$0.74_{0.12}$	$\mathbf{0.17}_{0.13}$	$23.37_{3.12}$	$0.78_{0.09}$	$0.12_{0.05}$	$\mathbf{25.47}_{2.73}$	$0.83_{0.07}$	$0.08_{0.03}$
	SIMPLI-8L	$\mathbf{23.10}_{3.09}$	$\mathbf{0.75}_{0.11}$	$\mathbf{0.17}_{0.12}$	$\mathbf{23.58}_{3.06}$	$\mathbf{0.79}_{0.09}$	$\mathbf{0.11}_{0.04}$	$\mathbf{25.47}_{2.64}$	$\mathbf{0.83}_{0.07}$	$\mathbf{0.07}_{0.03}$
8	IBRNet	$\mathbf{24.51}_{3.16}$	$0.77_{0.10}$	$0.18_{0.07}$	$\mathbf{23.98}_{3.31}$	$0.78_{0.09}$	$0.16_{0.06}$	$\mathbf{26.27}_{2.80}$	$0.83_{0.07}$	$0.11_{0.05}$
	LLFF	$21.22_{3.15}$	$0.59_{0.17}$	$0.28_{0.10}$	$22.91_{3.08}$	$0.77_{0.08}$	$0.17_{0.05}$	$24.29_{2.83}$	$0.78_{0.10}$	$0.14_{0.05}$
	DeepView†	$22.71_{3.60}$	$0.77_{0.11}$	$0.16_{0.09}$	$23.79_{2.49}$	$0.80_{0.07}$	$0.11_{0.03}$	$25.71_{2.75}$	$0.84_{0.07}$	$0.07_{0.03}$
	SIMPLI-4L	$24.02_{3.37}$	$0.78_{0.11}$	$0.16_{0.10}$	$23.86_{2.87}$	$0.80_{0.08}$	$\mathbf{0.10}_{0.03}$	$26.03_{2.36}$	$\mathbf{0.85}_{0.05}$	$0.07_{0.02}$
	SIMPLI-8L	$24.32_{3.37}$	$\mathbf{0.79}_{0.11}$	$\mathbf{0.15}_{0.10}$	$23.81_{2.66}$	$\mathbf{0.81}_{0.07}$	$\mathbf{0.10}_{0.03}$	$26.05_{2.29}$	$\mathbf{0.85}_{0.05}$	$\mathbf{0.06}_{0.02}$

Table 1. Results of evaluation on the hold-out test part of SWORD (30 scenes), RFF dataset (8 scenes) [24], and Shiny dataset (8 scenes) [46]. V denotes the number of source views. MPI representation for both LLFF and DeepView† consists of 40 planes. The dagger † indicates our re-implementation of the model. Note that the difference in metrics between our model and IBRNet is most often not significant due to the large std (indicated by subscripts), while SIMPLI produces a more compact representation, suitable for real-time rendering.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
IBRNet	$27.02_{2.09}$	$0.84_{0.05}$	$0.11_{0.03}$
DeepView	$\mathbf{29.52}_{2.92}$	$\mathbf{0.90}_{0.06}$	$\mathbf{0.05}_{0.01}$
DeepView†	$28.54_{2.35}$	$0.89_{0.04}$	$\mathbf{0.05}_{0.02}$
SIMPLI-4L	$27.73_{1.86}$	$0.87_{0.04}$	$0.07_{0.02}$
SIMPLI-8L	$28.01_{1.88}$	$0.88_{0.04}$	$0.06_{0.01}$

Table 2. Results of evaluation on the test part of Spaces dataset (10 scenes, large baseline) [7]. The number of source views equals 4. DeepView (original) and DeepView† (our modification) both employ 40 planes while SIMPLI produces only 4-8 layers. For this dataset one needs to trade off quality for representation compactness.

$\frac{P}{L}$ consecutive groups of equal size, and the depths of the planes are overcomposed within each group with predicted opacities. This produces L depth maps which do not intersected each other by design. Third, to obtain a mesh from each depth map, we treat each pixel of the map as a vertex and connect it to the six nearby vertices (top, right, bottom-right, bottom, left, top-left) with edges. Therefore, each quad (2×2 block of pixels) is converted into two triangle faces. The obtained multilayer representation is denoted as $fMLI_0$.

④ **Correction of MLI.** After the multilayer geometry $fMLI_0$ is obtained, we perform the correction step similar to ②, increasing the resolution of our representation to $H \times W$. Since the multilayer representation is relatively lightweight, it is possible to perform multiple error correction steps at this stage before obtaining the final representation. The number of such steps is denoted as τ. The influence of these additional steps is discussed in the ablation study below. The updated state $fMLI_\tau$ is converted from the feature domain to RGBA, and we refer to this final state as MLI. Examples of the resulting representations are presented in Fig. 1.

4.2. Error correction

Here we provide more details on the error correction procedure. It consists of computing the discrepancy with the input views and updating the representation based on this information. This procedure is performed similarly for MPI and MLI, therefore we assume the case of MPI for simplicity. Also we slightly abuse the notation in comparison with the previous subsection, as the described procedure does not depend on the exact step of our pipeline. The detailed schemes of all the steps are provided in Figs. S1 and S2.

Discrepancy computation. Let $fMPI$ (MPI in the feature domain) of the shape $P \times C \times h \times w$ be the input of the procedure. We use an *RGBA decoder* to predict RGB color and opacity from features. RGBA decoder is implemented with a fully connected network, applied to each of $P \times h \times w$ positions independently. The predicted color and opacity are concatenated with the original $fMPI$ along the channel axis before rendering on the source cameras. After rendering, we obtain tensors $\{\hat{F}_i\}_{i=1}^{V}$ of resolution $h' \times w'$, which is the target resolution for the error computation. Then we

compute the pixel-wise difference between the original feature tensor and the rendered view. These discrepancies are "unprojected" back onto the planes and serve as input to the aggregating block. Weights of the aggregating blocks and RGBA decoders are not shared between different steps of the pipeline.

In case of MPI, rendering is done with homography warping, while MLI requires the usage of a differentiable renderer [18]. Additionally, in case of MLI, the `unproj` operation "unprojects" the inputs on the predefined deformable layers instead of rigid planes. As the correction of MLI is done in full resolution, for the sake of memory consumption, we do not concatenate features to the predicted color before rendering and compute an error in the RGB domain only.

"Unprojection" on the deformable layers. In our pipeline the resolution of the depth maps and textures of layers are equal. Therefore, there exists a one-to-one correspondence between the vertices of the mesh layers and the texels with integer coordinates. To "unproject" a feature tensor, related to the certain source view, to the deformable layers, we project each vertex on that view using the pinhole camera model and take the feature value from the given tensor with bilinear grid sampling. This operation does not require us to use any differentiable renderer.

Aggregating modules. The aggregating module T_θ receives the current state $fMPI$ of shape $P \times C \times h' \times w'$ and the unprojected discrepancies PSV of the shape $V \times P \times (3 + C) \times h' \times w'$ (3 extra channels of PSV correspond to the output of the RGBA decoder). The self-attention module is applied to PSV along the V axis, following by the attention pooling with a single anchor.

The output of the pooling is concatenated with $fMPI$ channel-wise, as well as mean and variance calculated along the same V axis of PSV. The resulting tensor is passed through convolutional residual blocks [9], and each of the P planes is processed independently. The output is treated as a new state of $fMPI$.

At initialization ①, the aggregating module T_0 receives only the PSV, since there is no previous state of representation. Therefore, ResNet blocks operate on the output of the pooling step only. Also, as the resolution is low at this step, we employ 3D convolutional kernels, while at steps ② and ④ 2D convolutions are used.

4.3. Training.

The proposed system is trained on the dataset of scenes, unlike some prior approaches [25, 30, 38, 46] that require dedicated training per scene. The training process of the presented model is typical for NVS pipelines. First, we sample V source views and use them to construct an MLI representation, as explained above. Then we render MLI on holdout cameras sampled from the same scene and compare the generated images $\{I_j^n\}_{j=1}^N$ with the ground truth images $\{I_j^{gt}\}_{j=1}^N$ using the pixelwise ℓ_1 loss $L_1 = \frac{1}{3NHW}\sum_{j=1}^N \|I_j^n - I_j^{gt}\|_1$ as well as perceptual loss [13] $L_{\text{perc}} = \frac{1}{N}\sum_{j=1}^N \sum_t w_t \|VGG_t\left(I_j^n\right) - VGG_t\left(I_j^{gt}\right)\|_1$, where the weights w_t correspond to the different layers of the VGG network [34]. To smooth the geometry of the deformable layers, the total variation loss L_{tv} is imposed on the depth of each layer (the loss is computed for each of the L maps independently). Overall, our loss function is equal to $\lambda_1 L_1 + \lambda_{\text{perc}} L_{\text{perc}} + \lambda_{\text{tv}} L_{\text{tv}}$, and by default $\lambda_1 = 1$, $\lambda_{\text{perc}} = 2$, $\lambda_{\text{tv}} = 0.1$. We use Adam optimizer [16] with an initial learning rate equal to 10^{-4} and cosine annealing schedule [22]. More details can be found in our released code.

5. Experiments

5.1. Baselines, datasets and metrics

Baselines. To measure the performance of our system, we compare it with four baseline models: Stereo magnification (StereoMag) [50], LLFF [24], DeepView [7] and the more recent IBRNet [44]. StereoMag was designed for the extreme case of only two input views, so we trained this system on our data in this setting. The authors of LLFF have not open-sourced the training code for their network, therefore for the evaluation we use the inference code and checkpoint of this model provided by its authors.

The source code and checkpoints for the DeepView approach, which is most close in spirit to ours, were not released, while the authors admit the model is hard to implement [7]. However, they have shared the results of their model trained on the Spaces dataset [7]. Therefore, we compare our approach with theirs on this data. To make the comparison more fair, we train a special modification of our SIMPLI model called DeepView† (marked with dagger), which does not convert planes to layers and instead performs several error correction steps for the initial 40 planes.

Although the recently presented NeX model [46] outperforms LLFF and DeepView by a significant margin, we do not consider this approach as our baseline since it requires the full training procedure for each new scene which is far from our setting.

The authors of IBRNet evaluated its quality with and without fine-tuning for new scenes and concluded that their model shows the best results after additional per-scene optimization. However, for a fair comparison among methods, we do not fine-tune the IBRNet network and, therefore, measure the ability to generalize for all considered systems. We have tried to fit IBRNet on our training data, but obtained significantly worse results than the released pretrained model: PSNR on our holdout dataset equals 17.61

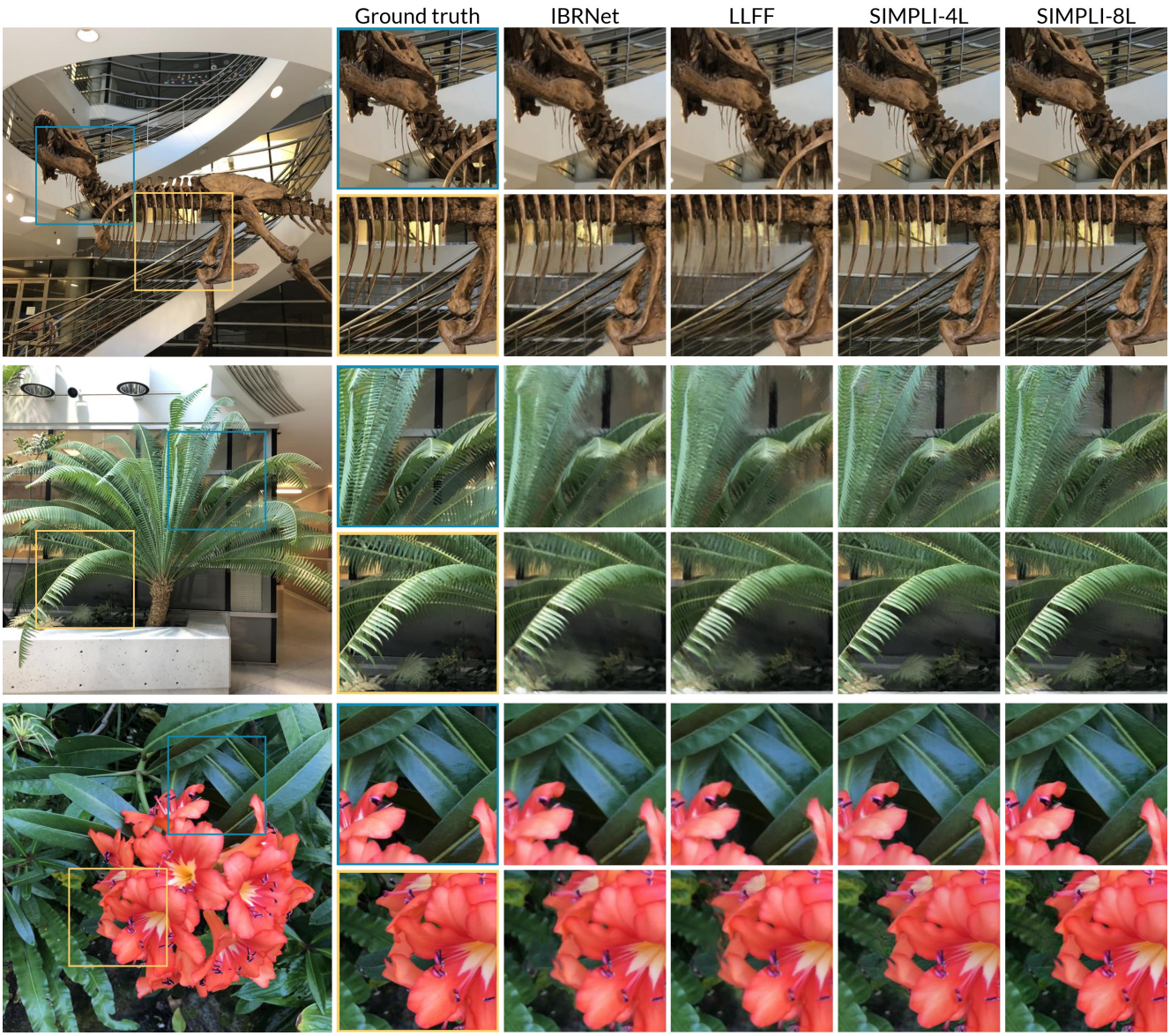

Figure 3. Comparison on a scene from the hold-out RFF dataset. In this experiment 8 source views were provided to all the models. Outputs of IBRNet and LLFF models are more blurry than results produced by SIMPLI (ours). As expected, using more layers in the MLI representation leads to better performance: note the bones of a T-Rex.

vs 22.69 for the released checkpoint. We suggest that this discrepancy is mostly due to the different sampling strategy of training camera poses, as well as more complex structure of our training dataset. Therefore, for the evaluation purposes, we stick to the pre-trained weights provided by the authors of IBRNet.

Training set. For training, we used 1,800 scenes from the training part of SWORD dataset [15]. The pipeline of data preparation was similar to that described in [50]. The source and novel poses are randomly sampled from the continuous range of frames within a scene, and the number of source images V varied from 2 to 8. To compare with DeepView, we also trained our models on the Spaces set (91 scenes).

Validation set. To validate various configurations of our model (see Sec. 5.2 for the list), we measured their quality on Real Forward-Facing data, referred to as RFF (40 scenes) [24] and a subsample (18 scenes) of data released with IBRNet paper, as well as Shiny dataset (8 scenes) [46].

Test set. Since IBRNet was trained on a mixture of datasets, including part of the RFF data and RealEstate10K [50], for the test set we selected 8 RFF scenes that IBRNet did not see and 8 scenes from the Shiny dataset [46]. We repeat the evaluation 20 times to compute the standard deviation. The

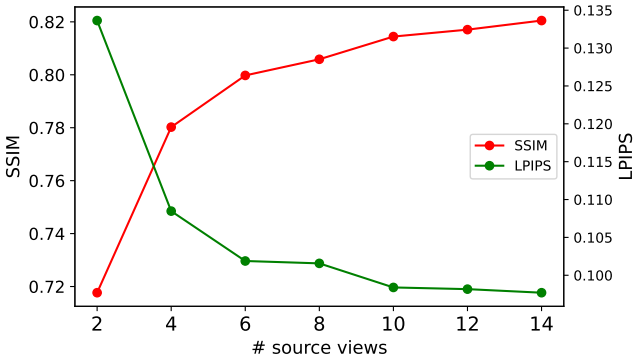

Figure 4. Increasing the number of source views results in better performance of the proposed SIMPLI model.

resolution of test images equals 384×512. Additionally, we report the results of different methods on 38 scenes from the test part of SWORD.

Sampling test views. We follow the evaluation protocol of DeepView, but extend it from photos taken from a camera rig to frames sampled from a short monocular video. Initially, a range of subsequent frames is chosen from the scene video, and afterward, source and novel cameras are sampled from this range without replacement.

In contrast, the authors of IBRNet selected the source camera poses for each novel pose as the nearest ones from the whole video to estimate the radiance field. Therefore, the scenario of DeepView looks more realistic for real-life applications, although this setup was not investigated for other baseline methods we have chosen. In addition, this setting also evaluates the robustness of different methods since it involves both the interpolation and extrapolation regimes.

Metrics. For evaluation, we employ several common metrics: perceptual similarity (LPIPS) [49], peak signal-to-noise ratio (PSNR), and structural similarity (SSIM). Before computing the metrics, we take central crops from the predicted and ground-truth images, preserving 90% of the image area, since the methods, based on MPI and MLI geometry, cannot fill the area outside the reference camera frustum [45]. To measure human preference, we showed pairs of videos to users of a crowdsourcing platform. Within each pair, a virtual camera was following the same predefined trajectory, and the assessors were to answer the question of which of the candidates looked more realistic (method known as 2AFC). For user study, entire set of RFF data (40 scenes), 18 scenes from the data released in IBRNet paper [44], and the Shiny dataset (8 scenes) were used. For each scene we generated two trajectories: rotation around the scene center (as in the IBRNet demo) and a spiral-like path with camera moving forward and backward. 800 different workers participated in the study, each pair was assessed by 40 of them.

Configuration	5 source views			8 source views		
	PSNR ↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓
w/o perceptual loss	$\mathbf{24.16}_{3.06}$	$\mathbf{0.79}_{0.09}$	$0.13_{0.06}$	$\mathbf{24.58}_{2.98}$	$\mathbf{0.80}_{0.08}$	$0.13_{0.05}$
w/o attention pooling	$23.67_{3.11}$	$0.77_{0.10}$	$0.12_{0.05}$	$24.09_{3.04}$	$0.79_{0.09}$	$0.11_{0.05}$
w/o error correction	$21.32_{4.15}$	$0.69_{0.12}$	$0.16_{0.05}$	$22.13_{3.12}$	$0.70_{0.11}$	$0.14_{0.05}$
$\tau = 0$ (④ omitted)	$23.15_{4.01}$	$0.76_{0.11}$	$0.12_{0.05}$	$23.92_{3.27}$	$0.78_{0.11}$	$0.11_{0.05}$
$\tau = 1$ (less steps)	$23.73_{3.16}$	$0.78_{0.10}$	$\mathbf{0.11}_{0.05}$	$24.16_{3.08}$	$0.79_{0.09}$	$0.11_{0.04}$
$\tau = 2$ (less steps)	$23.90_{3.13}$	$0.79_{0.09}$	$\mathbf{0.11}_{0.05}$	$24.30_{3.04}$	$\mathbf{0.80}_{0.08}$	$0.11_{0.04}$
$L = 2$ (fewer layers)	$23.41_{3.20}$	$0.77_{0.10}$	$0.12_{0.06}$	$23.76_{3.13}$	$0.78_{0.09}$	$0.12_{0.05}$
$L = 8$ (more layers)	$23.96_{3.10}$	$\mathbf{0.79}_{0.09}$	$\mathbf{0.11}_{0.05}$	$24.39_{3.00}$	$\mathbf{0.80}_{0.08}$	$\mathbf{0.10}_{0.04}$
default model	$23.79_{3.14}$	$0.78_{0.10}$	$\mathbf{0.11}_{0.05}$	$24.21_{3.05}$	$0.79_{0.09}$	$0.11_{0.05}$

Table 3. Ablation study. The default model contains $P = 40$ planes at intermediate steps and $L = 4$ layers in the final MLI representation. τ is the number of correction steps at stage ④, by default $\tau = 3$. Note that a model without correction steps both at stages ② and ④ demonstrates the worst performance. Despite removal of perceptual loss improves PSNR and SSIM, the results get blurry which leads to worse LPIPS value. Increasing the number of layers predictably raises the quality of results. Subscripts indicate the standard deviation.

Model	# params, mln	Building representation, sec	Rendering speed, fps
IBRNet	9.0	–	~ 0.4
LLFF	0.7	31.3	~ 60
DeepView†	1.7	65.4	~ 80
SIMPLI-4L	1.9	9.6	~ 200
SIMPLI-8L	2.0	10.6	~ 120

Table 4. Rendering speed for 8 source views at resolution of 768×1024. For all methods except IBRNet, we measure the time of representation building and rendering separately, as IBRNet requires forward pass of a neural network for each new frame, while other models may be used with graphics engines. DeepView† was implemented by us. Measurements are provided for NVIDIA P40 GPU.

5.2. Model configurations

The number of planes P in the preliminary MPI representation equals 40, and the default number of layers is $L = 4$, unless other values are explicitly specified, *i.e.* SIMPLI-8L denotes the architecture with $L = 8$ deformable layers. The modification called DeepView† does not convert planes to layers (*i.e.* step ③ is omitted), and step ④ is performed for MPI as well as ②.

We conducted an ablation study to evaluate the significance of different parts of our SIMPLI system and choose the best configuration. For this purpose, we trained versions of our model on two NVIDIA P40 GPUs for 90,000 iterations. Tab. 3 shows that an increase in the number of correcting steps can slightly improve quality. As expected, perceptual loss is important for the quality of high-frequency details measured by LPIPS. Using less layers worsens the model performance as well, while adding more layers or correction steps boosts the quality. This may be caused by the employed algorithm of converting planes to layers. We are going to explore alternative schemes to improve the performance of the SIMPLI model in future. Additionally,

Fig. 4 demonstrates that a greater number of source views fed to the model results in better quality.

5.3. Main results

To compare our approach with baselines, we train SIMPLI on 8 P40 GPUs for 500,000 iterations (approximately 5 days) with an effective batch size of 8. The main results are reported in Tab. 1 and Tab. 2. SIMPLI outperforms LLFF and typically is better than IBRNet, although in some cases obtaining slightly worse PSNR. This result may come from the fact that IBRNet was trained with ℓ_2 loss only, corresponding to PSNR (*c.f.* Tab. 3), while our model used perceptual loss. This is a clear advantage of our approach that it allows the training on patches large enough to use VGG-based losses, while IBRNet is trained on sparsely sampled pixels due to higher memory consumption. As reported in Tab. 4, SIMPLI allows for orders of magnitude faster rendering than IBRNet. While rendering in real time is also possible for LLFF, the quality it obtains is poor according to Tab. 1.

We observe that due to the large standard deviations, none of the models except DeepView outperforms SIMPLI by a statistically significant margin. Comparison of SIMPLI with DeepView† and original DeepView shows that there is a clear trade-off between quality and compactness on Spaces data, while on other datasets SIMPLI is consistently better. At the same time, SIMPLI provides a much more compact representation than DeepView: 4 layers against 40 planes.

Fig. 3 provides a qualitative comparison of different approaches, demonstrating that the proposed model has fewer artifacts in most cases. Although the metrics do not show a significant difference between the models considered, the study of human preference demonstrates the decisive advantage of SIMPLI over recently proposed baselines: our model achieves *81%* compared to LLFF and *79%* against IBRNet. See Appendix B and the accompanying video for more qualitative results.

5.4. Limitations

Although our model provides good quality of results in most cases, it still struggles from some limitations. First, similar to StereoMag, LLFF, StereoLayers and other methods which produce only the RGBA textures for the employed scene representation, SIMPLI cannot plausibly reconstruct view-dependent effects, *e.g.* specular reflections. One possible way to address this drawback is to predict the spherical harmonic coefficients instead of RGB color (*c.f.* NeX [46]), and we leave this for future work. Second, as shown in Fig. 5 for complicated scenes with fine-grained details, using as few as four layers is not enough. It is not immediately clear whether this effect should only be attributed to the small internal capacity of the representation with a few layers. Instead, the suboptimal procedure for converting planes to layers (step ③) may be to blame. Therefore, searching for a more appropriate merging algorithm, as well as extending the error correction step to the depth maps, are other ways to improve our model.

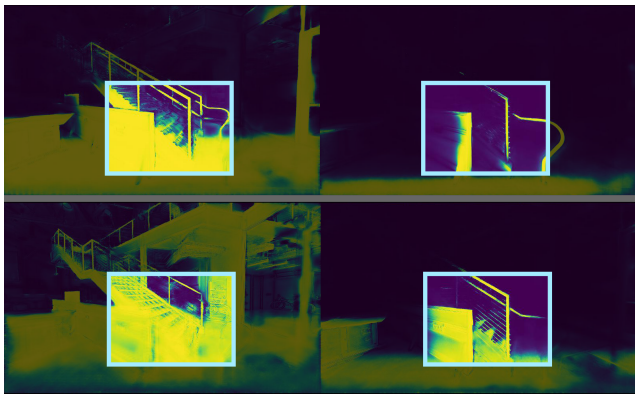

Figure 5. Example of the limitations. The opacity channel of predicted textures for the layers in the depth range corresponding to the ground truth position of the stair railing. Left: model with 4 layers, right: model with 8 layers. Obviously, when using too few layers, the proposed SIMPLI is not capable of faithfully representing thin objects.

6. Conclusion

We presented a method of novel view synthesis called SIMPLI. It employs a mesh-based scene representation that consists of a set of non-intersecting semitransparent layers. RGBA textures for the mesh layers are inferred with a multistage neural pipeline that refines the representation. Our network is trained on a dataset of real-life scenes and generalizes well to the unseen data without fine-tuning, allowing for on-device usage.

Our approach extends existing multiview methods in three major ways: (i) we propose to use an adaptive and more compact representation of the scene (4 or 8 layers instead of 40 planes), (ii) we use input frames from the natural handheld video instead of the calibrated rig, and (iii) the presented system does not rely on the predefined number of input views, instead being able to work with an arbitrary number of source images.

Evaluation shows that SIMPLI outperforms recent state-of-the-art methods or produces results of similar quality, while excelling over them in the speed of rendering or the size of the obtained scene representation.

Acknowledgements

The authors thank V. Aliev, A.-T. Ardelean, A. Ashukha, R. Fabbricatore, A. Kharlamov, V. Lempitsky, and R. Suvorov for their comments, which greatly improved the manuscript.

A. Additional details

Architectures. Detailed diagrams of the steps of our pipeline are shown in Fig. S1. The architectures of the aggregating modules are described in Fig. S2.

MLI vs LDI. Since the seminal work on layered depth images (LDI) [32], multiple papers considered equipping each pixel of the reference image with a stack of depth values. However, the original definition of this representation [32] assumed that the size of this stack may differ between pixels. In addition, any connections between pixels were not imposed. Although later manuscripts introduced explicit local connectivity of neighboring pixels [33], we stick to another terminology and refer to our representation as a layered mesh [2] or a multilayer image (MLI) [15]. The latter name is preferred, as it reveals the relation to multiplane images [50]: just like them, our proxy geometry contains semitransparent RGBA textures. On the contrary, many methods that have been reported to employ LDI representation do not use the opacity channel [6, 42, 33, 17]. Besides, MLI contains a predefined number of layers; therefore, each pixel gets the same number of depth values. While the layers are non-overlapping by design, a ray from a novel camera can intersect each layer in several points, justifying the usage of z-buffer during the rasterization step. In contrast, the original LDI representation did not need the z-buffer, and McMillan's warp ordering algorithm was used instead.

B. Additional results

Fig. S3 demonstrates the results of our SIMPLI method in the case of 4 layers. Also we provide an additional comparison with the baseline methods on publicly available datasets [44, 24]. We show visual results for 2 input views in Fig. S4, for 5 input views in Fig. S5 and for 8 – in Fig. S6. These results correlate with the metrics reported in the main text. For two or five input views, our method clearly outperforms all baselines and produces the most visually pleasant results. Also, most of the crops for the eight input images show that our method is at least on par with existing approaches. Note that all the demonstrated crops and scenes are uncurated.

Fig. S7 demonstrates a comparison of our model with the DeepView system [7] on the Spaces dataset [7]. We show the results for the small and large camera baselines separately. As may be seen from the figures, our model produces slightly blurrier results than DeepView does. However, it allows us to get a much more compact scene representation, as was discussed in the main text. We demonstrate the visual comparison for SIMPLI with a different number of layers in the MLI representation in Figs. S4 to S6. We observe the degradation of the quality with decreasing of the number of layers in the final representation.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, 2022. 1

[2] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. In *ACM TOG*, 2020. 1, 2, 9

[3] 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*, 2021. 2

[4] 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*, 2021. 2

[5] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *ACM TOG*, 1996. 2

[6] Helisa Dhamo, Keisuke Tateno, Iro Laina, Nassir Navab, and Federico Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 125:333–340, 2019. 2, 9

[7] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Styles Overbeck, Noah Snavely, and Richard Tucker. Deepview: High-quality view synthesis by learned gradient descent. In *CVPR*, 2019. 1, 2, 3, 4, 5, 9

[8] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *ACM TOG*, 1996. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, June 2016. 5

[10] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. In *ACM TOG*, 2018. 1

[11] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. In *ACM TOG*, 2016. 2

[12] 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*, 2021. 1, 2

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 5

[14] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. In *ACM TOG*, 2016. 2

[15] Taras Khakhulin, Denis Korzhenkov, Pavel Solovev, Gleb Sterkin, Timotei Ardelean, and Victor Lempitsky. Stereo magnification with multi-layer images. In *CVPR*, 2022. 1, 2, 6, 9

[16] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 5

[17] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Peter Vajda, Ayush Saraf, and Michael Cohen. One shot 3d photography. In *ACM TOG*, 2020. 9

[18] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 2020. 5

[19] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019. 3

[20] 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*, 2020. 1, 2

[21] 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 *CVPR*, 2022. 1

[22] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*, 2017. 5

[23] F Landis Markley, Yang Cheng, John L Crassidis, and Yaakov Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007. 3

[24] 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. In *ACM TOG*, 2019. 1, 2, 4, 5, 6, 9

[25] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 5

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *ACM TOG*, 2022. 2

[27] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. 1

[28] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. In *SIGGRAPH Asia*, 2017. 2

[29] Thomas Porter and Tom Duff. Compositing digital images. In *ACM TOG*, 1984. 3

[30] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 5

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

[32] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *ACM TOG*, 1998. 2, 9

[33] M. L. Shih, S. Y. Su, J. Kopf, and J. B. Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2, 9

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 5

[35] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 1

[36] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[37] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021. 2

[38] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. In *ACM TOG*, 2019. 5

[39] Alex Trevithick and Bo Yang. GRF: Learning a general radiance field for 3D scene. In *ICCV*, 2021. 2

[40] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 1

[41] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 2

[42] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 9

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 3

[44] 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*, 2021. 1, 2, 5, 7, 9

[45] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 7

[46] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-Time View Synthesis With Neural Basis Expansion. In *CVPR*, 2021. 2, 4, 5, 6, 8

[47] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 1

[48] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural Radiance Fields From One or Few Images. In *CVPR*, 2021. 2

[49] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[50] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *ACM TOG*, 2018. 1, 2, 3, 5, 6, 9

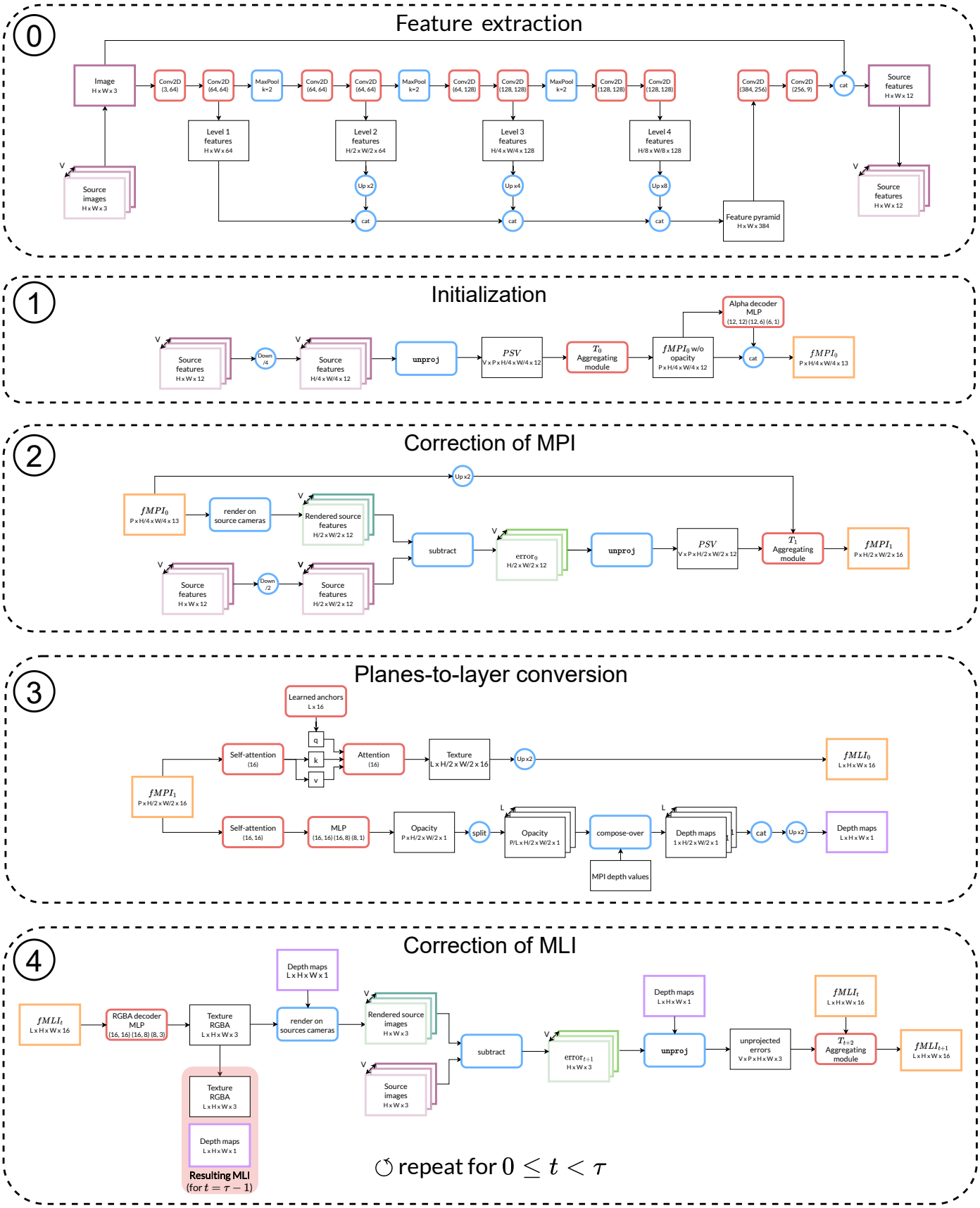

Figure S1. The detailed diagram of our pipeline. Please zoom in for details.

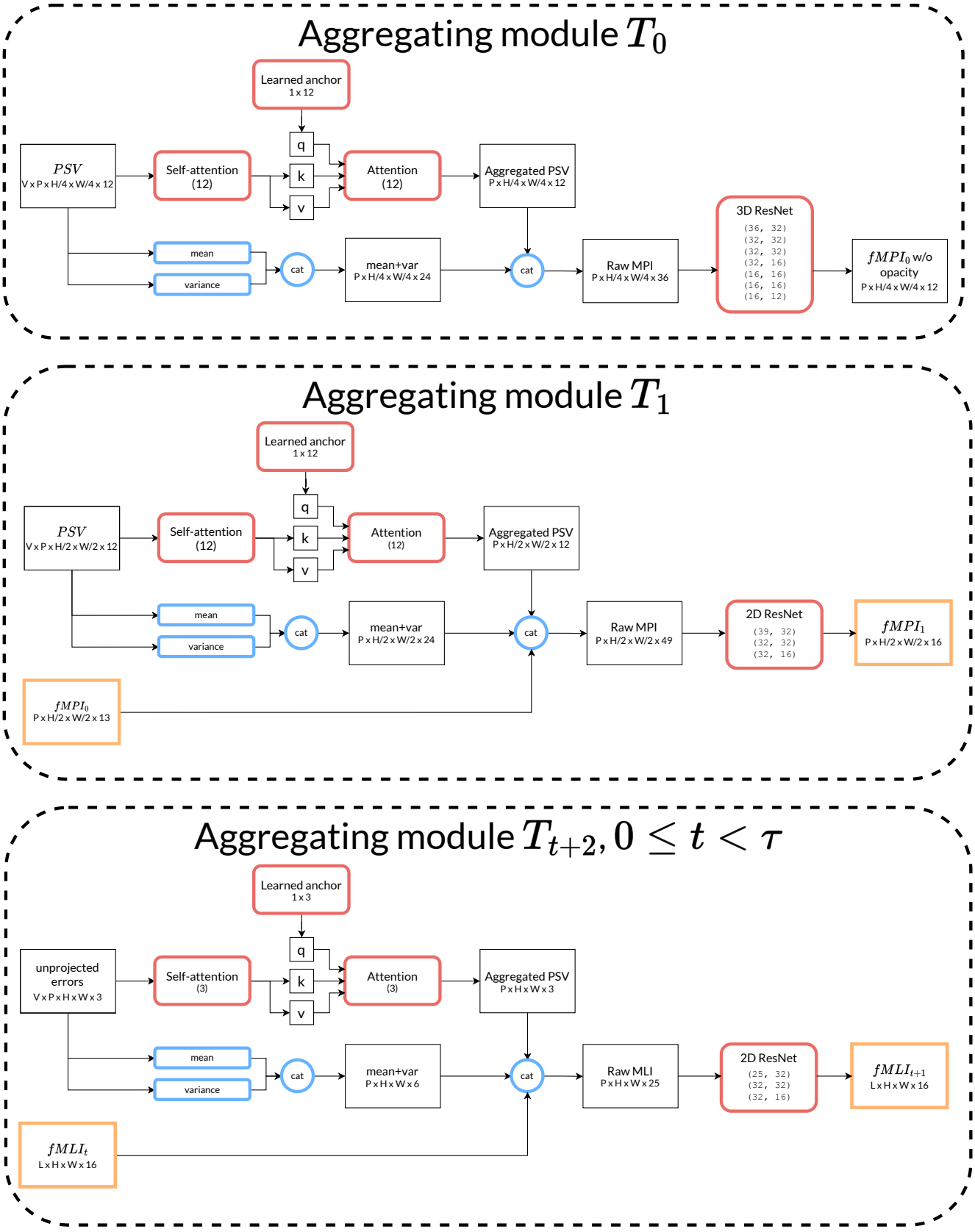

Figure S2. Architecture of aggregating modules. Please zoom in for details.

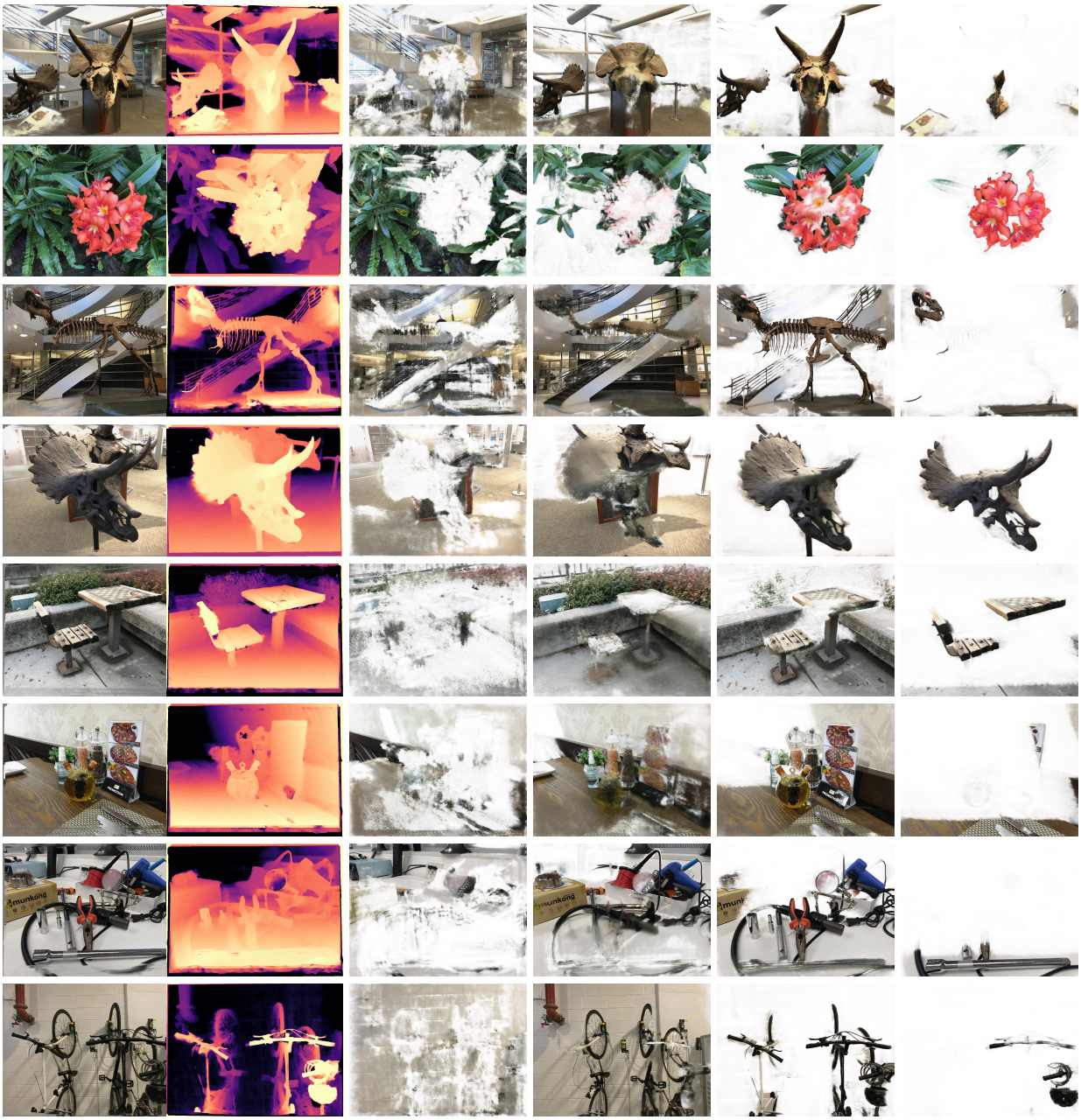

Figure S3. Extension of Fig. 1. The textures of MLI representation with 4 deformable layers estimated by SIMPLI. Left to right: generated novel view, corresponding depth map, four semitransparent textures in the back-to-front order. The inferred depth map is computed by overcomposing the per-layer depth maps w.r.t. the opacity extracted from the corresponding RGBA textures.

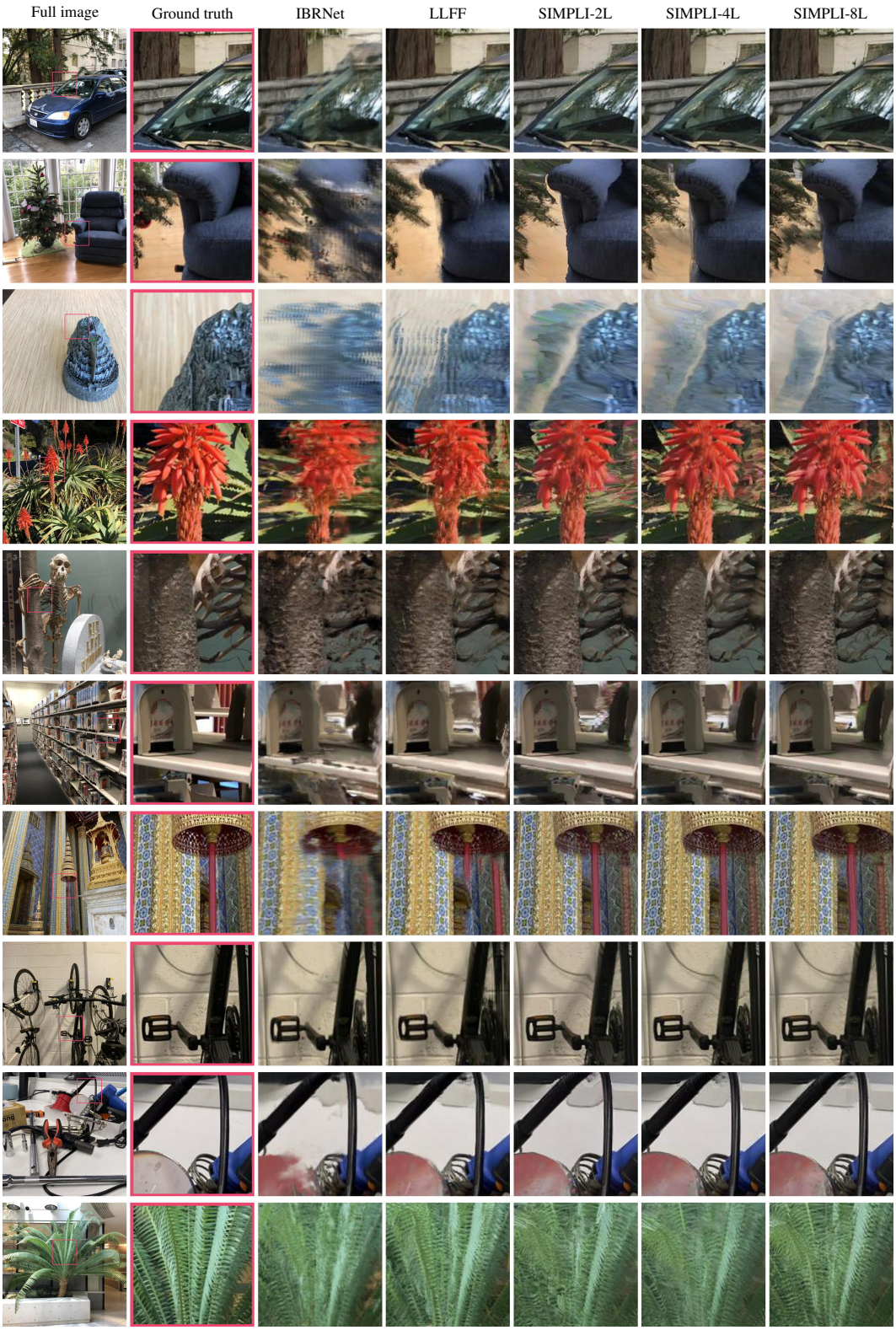

Figure S4. Results for real novel cameras with **two** input views given. Note that IBRNet cannot produce any information for areas unobserved from the source views.

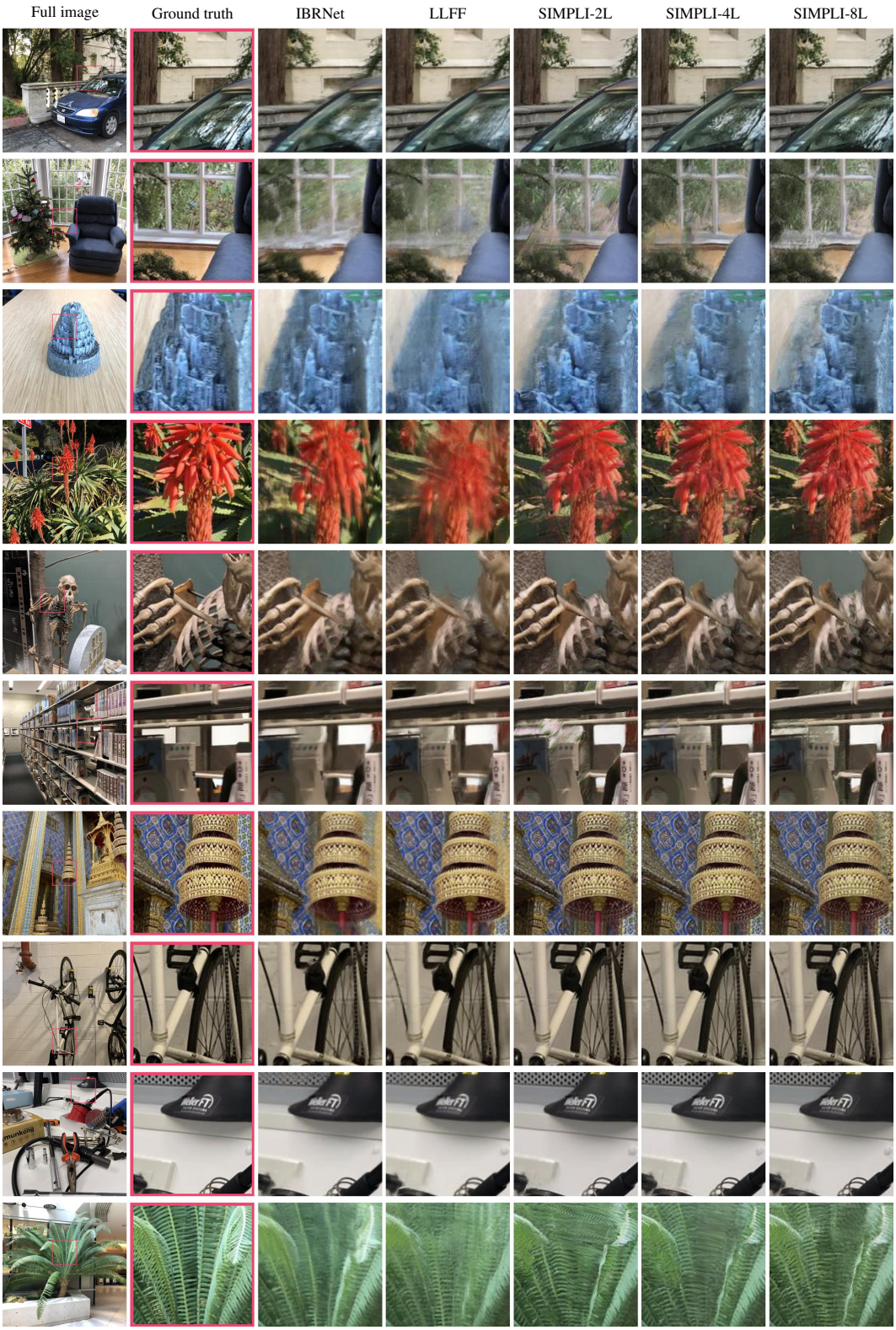

Figure S5. Results for real novel cameras with **five** input views given. The outputs of SIMPLI-8L are the most similar to the ground truth frames and have less artifacts than other models.

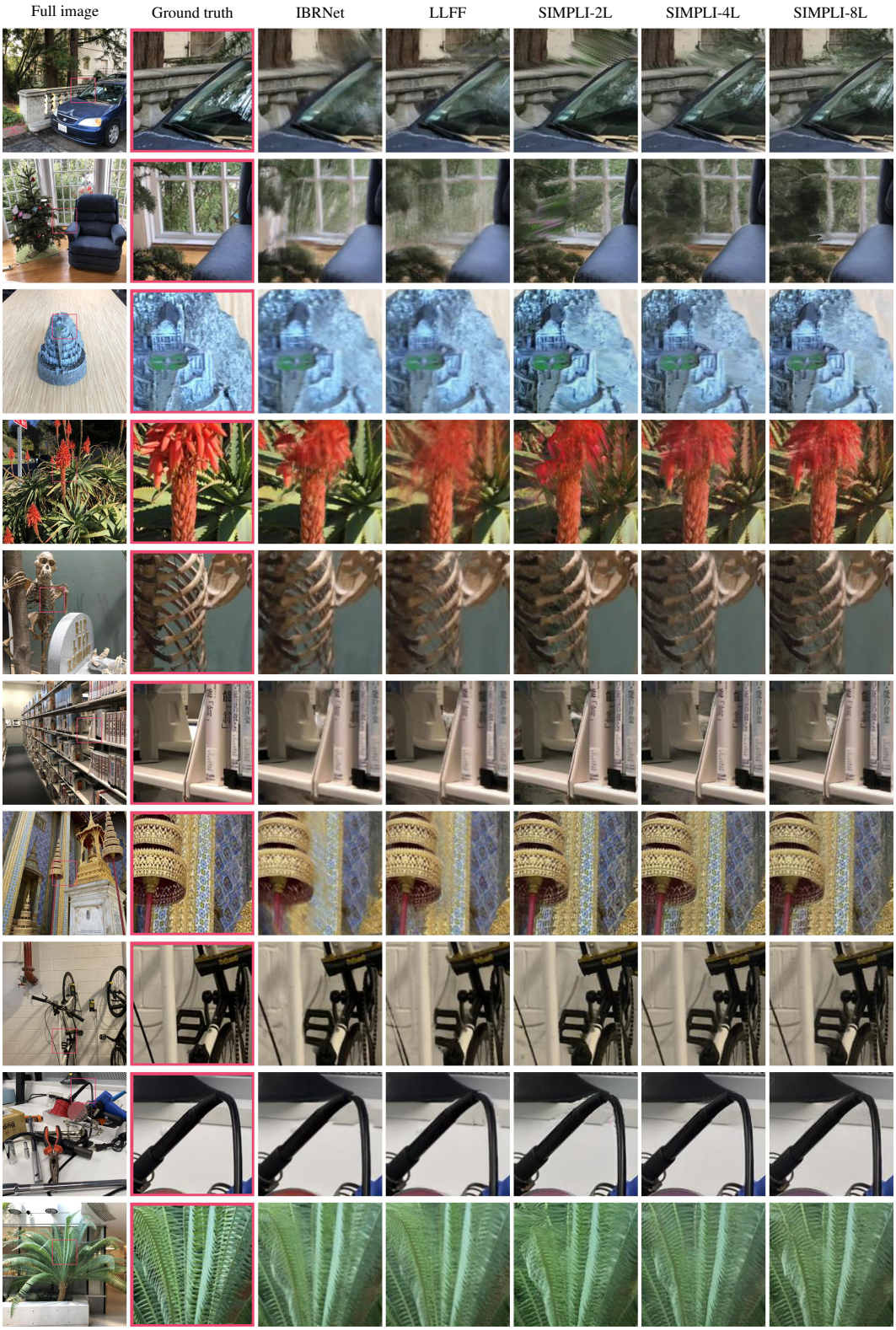

Figure S6. Results for real novel cameras with **eight** input views given. This is the most competitive scenario. Note that both IBRNet and LLFF tend to produce many artifacts, *e.g.* blurriness, while frames rendered by SIMPLI are more sharp.

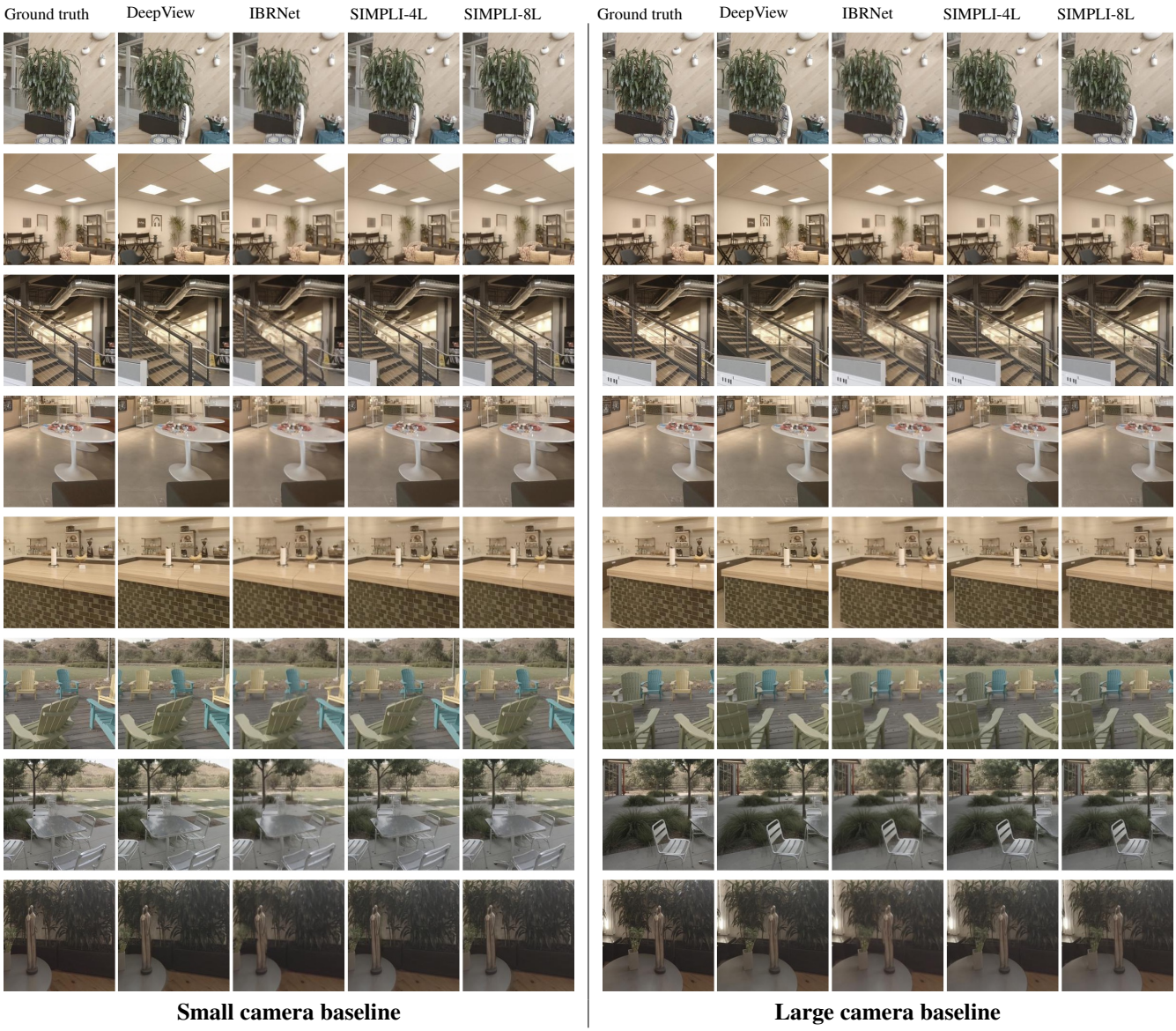

Figure S7. Results for the Spaces dataset for DeepView (40 planes) and SIMPLI with 4 and 8 layers. Our model produces more blurry and less bright results, trading off for more compact representation of the scene.