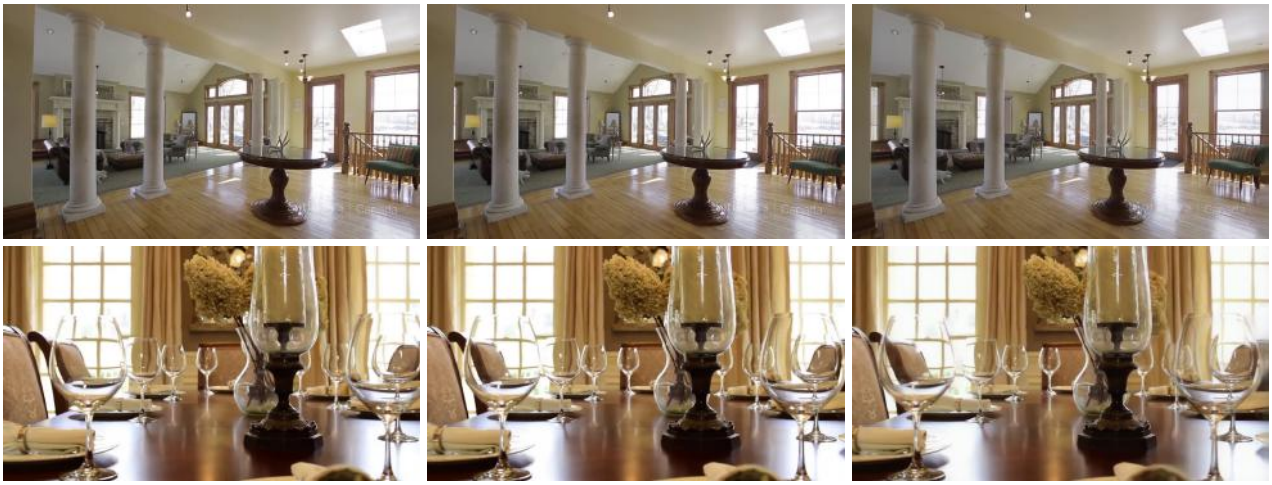
FaDIV-Syn: Fast Depth-Independent View Synthesis

Andre Rochow
University of Bonn
rochow@uni-bonn.de

Max Schwarz
University of Bonn
schwarz@ais.uni-bonn.de

Michael Weinmann
University of Bonn

Sven Behnke
University of Bonn



Input 1 Input 2 Inter-/Extrapolated

Figure 1: FaDIV-Syn inter- and extrapolates images from two views. Input images from RealEstate10k [48] test set. This figure is animated—if the videos are not visible we refer to our supplementary material.

Abstract

We introduce FaDIV-Syn, a fast depth-independent view synthesis method. Our multi-view approach addresses the problem that view synthesis methods are often limited by their depth estimation stage, where incorrect depth predictions can lead to large projection errors. To avoid this issue, we efficiently warp multiple input images into the target frame for a range of assumed depth planes. The resulting tensor representation is fed into a U-Net-like CNN with gated convolutions, which directly produces the novel output view. We therefore side-step explicit depth estimation. This improves efficiency and performance on transparent, reflective, and feature-less scene parts. FaDIV-Syn can handle both interpolation and extrapolation tasks and outperforms state-of-the-art extrapolation methods on the large-scale RealEstate10k dataset. In contrast to comparable methods, it is capable of real-time operation due to its lightweight architecture. We further demonstrate data efficiency of FaDIV-Syn by training from fewer examples as well as its generalization to higher resolutions and arbitrary depth ranges under severe depth discretization. Source code will be made available.

1. Introduction

Novel view synthesis aims to estimate images from novel viewpoints of a scene captured from one or more reference views. It has applications in virtual reality, 3D movies, or any other field where free choice of viewpoint is desired. The task is challenging as scene geometry and surface properties are not given, but have to be inferred from the available views. Additionally, viewpoint changes induce both occlusions, where foreground objects occlude previously visible backgrounds, and disocclusions, which uncover previously invisible backgrounds. In the latter case, the disoccluded content has to be guessed from its context. The view synthesis task is related to Multi-View Stereo (MVS), which aims to recover scene geometry only.

Here, we focus on the problem setting of real-time view interpolation and extrapolation from two RGB images, which show the scene from roughly the same direction—as, for example, when captured from a stereo camera. We note, though, that our method is applicable to more input views.

Many stereo [14, 30, 13, 5] and view synthesis methods [48, 36, 15] use Plane Sweep Volumes (PSV) [7], which warp the input views on a range of planes defined in the target camera and thus pre-transform the input data under

the assumption of a range of discrete depths. Usually, a disparity or depth map is estimated from the PSV, which is then used to project the input views into the target frame correctly [15] or to generate representations such as multiplane images [48, 36]. However, this approach creates a bottleneck: Imprecise or wrong depth estimates, which occur especially on uniform, transparent, or reflective surfaces, will result in loss of information and lead to failures later on in the synthesis pipeline.

Our proposed method is related to Image-Based Rendering approaches, but forgoes the geometry estimation step by operating on the PSV directly to compute the output RGB image, without computing explicit depth. To this end, we learn an RGB generator network which processes the PSV to directly synthesize the novel view and equip it with operations such as group- and gated convolutions [8, 45], which are suitable for detecting layer-wise correspondences, masking of irrelevant areas, and blending. Unlike most Image-Based Rendering (IBR) approaches, FaDIV-Syn has no explicit blending or inpainting stage, which allows distribution of the blending and inpainting operations throughout the learned network at appropriate abstraction levels. Only a single forward pass is required for view interpolation and extrapolation. In summary, our contributions include 1) a real-time view synthesis network operating on plane sweep volumes and 2) a detailed evaluation on the large-scale RealEstate10k dataset [48], where we outperform comparable methods in accuracy and runtime.

2. Related Work

A large variety of novel view synthesis approaches have been developed. For a broader review, we refer to Nguyen et al. [21] or Tewari et al. [37].

Image Based Rendering (IBR). In contrast to classical rendering of 3D scenes using textured geometry, IBR methods aim to render novel views by combining input images in the target pose [25, 12, 11, 3, 41, 15, 21, 43, 38, 27, 28]. To be able to project the input images correctly, IBR methods still require geometry, often in the form of depth maps, which are either available or estimated. Recent approaches use blending to combine the images [25, 12, 11, 27]. Hedman et al. [12] learn the blending operation end-to-end. Going further, Riegler and Koltun [27] use a recurrent blending decoder in order to deal with a varying number of input images. In later work [28] heuristic input image selection is replaced with a fully-differentiable synthesis block. Penner and Zhang [25] introduce soft visibility volumes, which encode occlusion probabilities and thus avoid early decisions, but require a larger number of input views to compute. Kalantari et al. [15] synthesize novel views in light field datasets. They use the corner cameras to predict depth in the target view, which is then used to warp the input views. Nguyen et al. [21] introduce RGBD-Net, which first estimates depth using a multi-scale PSV, warps the input images into the target frame, performs explicit blending, and refines the warped image using a depth-aware network. Our method also works directly on the input images, but does not compute or require depth explicitly. Blending is learned implicitly by the network, together with detection and inpainting of extrapolated/disoccluded regions.

Geometry-Based Approaches. Recent approaches [1, 6, 42, 35] use depth features to spatially project pixel information and refine these projections to a target view. Wiles et al. [42], Chen et al. [4] process single input images, estimating monocular depth in an end-to-end fashion. Wiles et al. [42] implement a differentiable point cloud renderer that allows z-buffering and splatting. Chen et al. [4] predict depth in the target view using a transforming auto-encoder, that explicitly transforms latent code before entering the decoder. Similarly, Olszewski et al. [23] learn implicit voxel representations and transform encoded representations explicitly. Srinivasan et al. [35] predict RGB-D light fields from a single RGB image. They estimate precise scene geometry, render it to the target frame, and predict occluded rays using a second CNN. Extreme View [6] predicts depth probabilities along camera rays in multiple input images and unites them in the target camera pose. They discretize the number of possible depth values and therefore reduce the depth estimation problem to a classification problem. A more recent approach [10] learns novel view synthesis without target view supervision by performing two synthesis steps, initially to an arbitrary target pose and from there to a pose where ground truth is available. In contrast to these methods, FaDIV-Syn does not feature an explicit geometry representation. We argue that explicit geometry—besides requiring more effort to compute—forces early resolution of ambiguities, which can lead to loss of information.

Multiplane and Layered Depth Images (MPIs/LDIs). A multiplane image consists of multiple depth planes, which store RGB and alpha values. Once computed for a set of input images, novel views can be synthesized very efficiently by warping and blending the individual layers. One can attempt to predict MPIs from single input images [17, 40]. Tucker and Snavely [40] train a network to estimate scale-invariant depth and require additional sparse point clouds to recover scale. Multiview approaches [36, 48, 18] use information from additional camera poses to place surfaces at the correct MPI layer. Plane sweeping [48] or warping [36] at different depths creates a suitable representation for the network. Mildenhall et al. [18] blend the layers of multiple MPIs to generate novel views with local light fields. Recently, Attal et al. [2] extended the key idea of multiplane images to multisphere images, to synthesize 360° images in real-time, although at lower resolution.

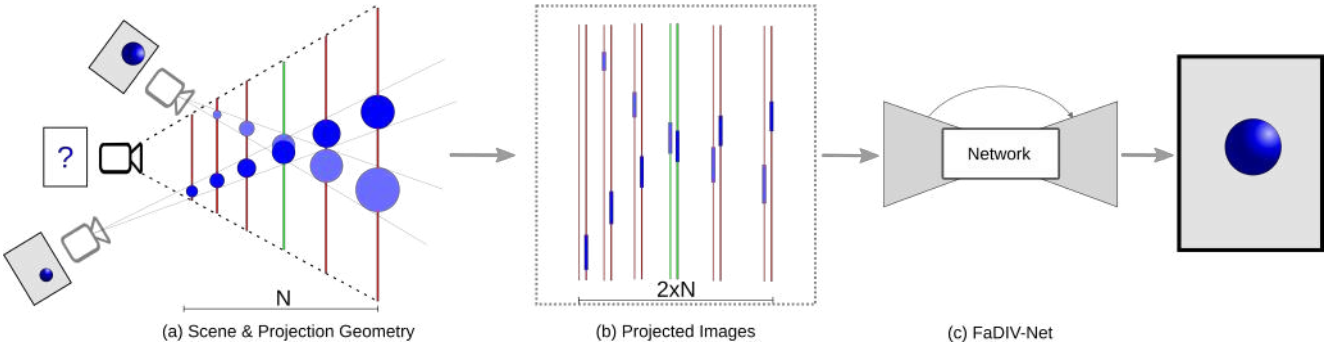


Figure 2: The FaDIV-Syn architecture. (a) Input images (gray) are projected into the target camera (black) for each depth plane (red/green) defined in the target frame. For a particular surface in the scene (blue circle), there will be a depth plane where the projections most closely align (green). This plane corresponds to the true object depth. (b) The resulting projected images are stacked and fed into the view synthesis network (c), which directly predicts the target image.

Other approaches [34, 32, 50] build on Layered Depth Images (LDI) [31], which store multiple RGB and depth values per pixel. Snavely et al. [34] train a CNN to predict a two-layered LDI, where the network learns to predict occluded pixels. Shih et al. [32] use an LDI representation to turn a single image into a 3D photo by inpainting color and depth of the occluded areas. While inspired by MPI approaches, FaDIV-Syn bypasses MPI generation and instead determines a novel view from multiple warped planes directly and is applicable for dynamic scenes in real time.

Neural Rendering. Very recently, view synthesis approaches based on Neural Radiance Fields (NeRF) [19] have been introduced, which employ a neural network as a learnable density and radiance function over the scene volume. Novel views can be synthesized using classical volume rendering techniques. The sub-sequent improvements [46, 44, 41, 24] show impressive results on a variety of scenes. However, with the exception of Wang et al. [41], NeRFs have to be trained on the target scene, making them unsuitable for dynamic scenes, and are typically given more than two input images. While methods designed for dynamic scenes exist [24, 26, 9, 39], they require offline training or processing phases as well.

3. Proposed Method

The key idea of FaDIV-Syn is to preprocess and transform the input images into the target frame, without losing information. Of course, the transformation requires depth information. Instead of estimating depth, we sample multiple depth variants (see Sec. 3.2) and present the resulting possibilities to the network. The induced representation is well-suited for the view synthesis task.

While in principle FaDIV-Syn can operate with any number of input images, we present and evaluate the method for two input images (in the following called I_1 and I_2) and one output image (I_O). For both input cameras, we sample N depth planes and for each plane assume that the entire image lies on it. When projecting these planes into the target view, one could, for $N \to \infty$, determine each pixel's 3D position by searching for correspondences in the warped layers (see Figs. 2 and 4). When performing the correspondence estimation and merging task with a learned network, we can reduce N to a small number, since the network can learn to interpolate between planes with adjacent depth levels.

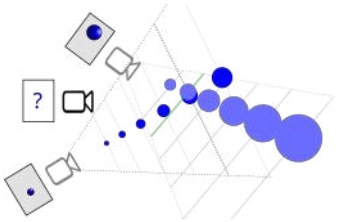


Figure 3: When defining depth planes depending on the individual camera's coordinate system (compare against Fig. 2), correspondences end up on different depth planes, making the representation harder to process.

3.1. Plane Sweep Volume

Planar geometry is especially well-suited for camera-to-camera projection, since the resulting warping operation can be done efficiently. A naïve approach might be to define depth planes in both input images I_1 and I_2. This corresponds to computing two PSVs and then attempting to merge them. However, this is not a well-designed representation, since corresponding elements might be on very different depth planes (see Fig. 3).

Instead, we define the planes in the target image I_O (see Fig. 2), as it is commonly done in plane sweeping multi-view stereo approaches [7]. For each plane i, we define $P_k^{(i)} \in P$ as the image resulting from projecting I_k onto the plane, and then into I_O. Using this representation, we can

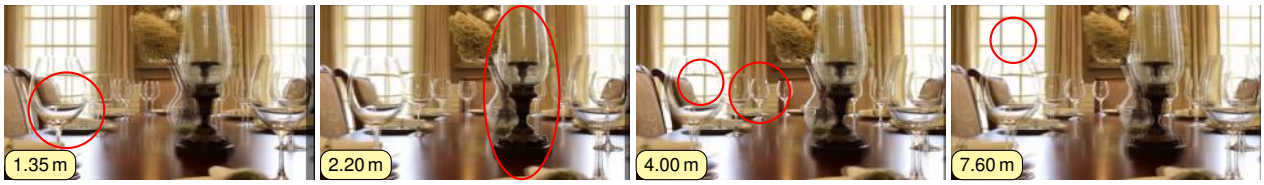


Figure 4: Plane sweep volume (PSV). Shown are four planes of the Fig. 1 scene with α-blended projections of the two input images. Note that RealEstate10k has only estimated global scale, so the given plane distances are only accurate up to scale.

define a "hard-wired" view synthesis method f:

$$f(I_1, I_2, p) = \begin{cases} P_1^{(D(p))}(p) & \text{if } p \text{ visible in } I_1, I_2 \\ g(I_1, I_2, p) & \text{otherwise ,} \end{cases} \quad (1)$$

$$D(p) = \arg\max_j Q(P_1^{(j)}, P_2^{(j)}, p), \quad (2)$$

where $p = (x, y)$ is a pixel in the target image I_O, g is an inpainting method, D is the (internal) depth estimate, Q is a correspondence quality estimator, and j denotes the plane with optimal correspondence. Note that only images in P of the same depth need to compared. If the planes were not aligned (Fig. 3), we would need to compute $\arg\max_j(\max_i(Q(P_1^{(j)}, P_2^{(i)}, p))$, which is obviously more complex and harder to learn. Figure 4 shows an exemplary PSV where the idea presented in Eq. (1) will be immediately apparent. Since perfect Q and g are not known, we will learn a CNN approximating f.

Given a plane $L_O^{(j)} = (0 \quad 0 \quad -1 \quad d_j)^T$ defined in the target frame I_O, we can find the plane in frame I_i as:

$$L_i^{(j)} = {}^{O}T_i{}^{-T} \cdot {L_O^{(j)}}^T, \quad (3)$$

where ${}^{O}T_i$ is the homogenous transformation between output and input frame i. We warp images efficiently using the resulting inverse homography.

3.2. Depth Discretization

In real world scenarios, the depth relative to the camera lens can take any positive value $d \in \mathbb{R}_{>0}$. For increasingly distant objects, the spatial error of projected pixels caused by wrong depth decreases. Therefore, it is legitimate to set a maximum depth. Our main network uses only 17 depth planes, where we distribute the planes uniformly in disparity space for the depth range [0.3,8] m and one additional background plane at 16 m. Compared to related work [36, 48], this discretization is quite severe. The RealEstate10k dataset, in turn, contains a large number of drone-captured scenes with very large distances but also close geometry, which makes the task even more difficult.

3.3. Generator Network

When projecting an image according to a known depth map, the disocclusion areas requiring inpainting are those

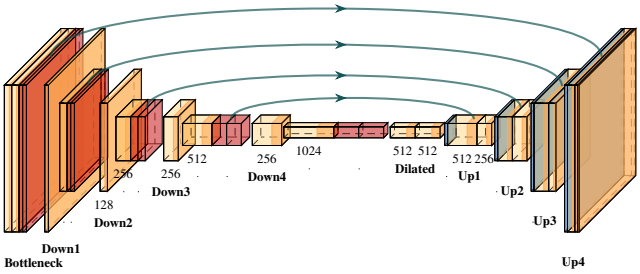


Figure 5: FaDIV-Net base architecture without group convolution stage. Yellow blocks denote Conv2D+BN+ReLU, red blocks show gating, and blue blocks transposed convolutions. For a more detailed description, please refer to the supplementary material.

where no pixel from the image is mapped to. When considering a PSV, disocclusion areas are no longer trivial to determine. Thus, our network must learn to distinguish between (1) areas of sufficient correspondences in the warped layers, (2) areas of no correspondence but sufficient correspondence in different layers and (3) areas of disocclusion and occlusion. Hence, the network must learn under the constraint of geometric consistency to fuse and correct sufficiently corresponding areas from warped layers and recognize inpainting areas to fill them with realistic content. In contrast to Thies et al. [38], who also learn implicit blending, we avoid early decisions in a depth estimation stage, which may lead to quality degradation later in the pipeline.

Our main network is based on a U-Net [29] architecture as shown in Fig. 5. It consists of 4 downsampling and 4 upsampling blocks with skip connections and a dilated convolution in the middle. For upsampling we use transposed convolutions. Each downsampling block consists of two convolutions with batch normalization and ReLU activation. The second convolution has a stride of two for downsampling. Afterwards, we split the feature dimension in half and multiply the two parts to realize gating, as explained directly below. Since the network is supposed to process high-resolution input images of dimension $2 \times N \times 3 \times H \times W$, we send the entire input at the first gated convolution through a bottleneck. This way, the network can eliminate non-corresponding and combine corresponding areas. For further details on the network architecture we refer to the sup-

plementary material.

Gated Convolutions. Gated convolutions [8] have recently shown promising results in image inpainting [45]. Instead of $C(W_f, I) = \sigma(W_f \circledast I)$, a gating layer calculates the non-linear output

$$GC(W_f, W_g, I) = \sigma_1(W_f \circledast I) \odot \sigma_2(W_g \circledast I), \quad (4)$$

where $\odot$ denotes the elementwise multiplication, $\circledast$ is the convolution. W_g, W_f are two different convolutional filters, and $\sigma_{1,2}$ is the ReLU function in our approach. The formula shows that gating directly influences the actual feature extraction and can thus adapt it to the context. Gating layers can thus help the network especially with performing masking-like operations (e.g. when recognizing corresponding depth planes), performing blending, or determining inpainting areas [45].

3.4. Extended Architecture

Aligning the depth layers allows us to further extend the network. The alignment ensures that corresponding areas only appear in corresponding planes (see Fig. 2). Hence, we preprocess the corresponding planes by a gated group convolutional layer (GGC) before they enter the base network. This can be done under the assumption that other layers are initially irrelevant to the considered pair. Grouping saves processing time and makes it easier for the network to comprehend the context of the task to be learned. Additionally, the alignment allows us to pre-compute a pairwise correspondence metric between the layers. We concatenate a structural similarity [49] map to each corresponding layer pair, which can be used by the network as an aid. Especially gated convolutions can help the network to understand the connections between the similarity maps and the task to be accomplished in these areas, as Yu et al. [45] showed in the context of image inpainting. Furthermore, one can enhance the receptive field of the network in the group convolutional part if necessary, which is more time-efficient than in the base network.

3.5. Training

The network is trained in a supervised manner from a triple $(I_1, I_2, \tilde{I}_O)$ with known camera poses and intrinsics. We define the loss function

$$\mathcal{L}(I_O, \tilde{I}_O) = \lambda_1 \mathcal{L}_1(I_O, \tilde{I}_O) + \lambda_p \mathcal{L}_{perceptual}(I_O, \tilde{I}_O), \quad (5)$$

where the perceptual loss is based on a VGG-19 [33] network Ψ pretrained on ImageNet and defined as

$$\mathcal{L}_{perceptual}(I_O, \tilde{I}_O) = \sum_l^L \frac{w_l}{N_{\Psi_l}} |\Psi_l(I_O) - \Psi_l(\tilde{I}_O)|, \quad (6)$$

where $\Psi_l(\cdot)$ is the activation of the l-th layer, w_l is a weight factor of the l-th layer and N_{Ψ_l} are the number of elements in the l-th layer. We train the network with a batch size of 20 and the Adam optimizer with $\beta_{1,2} = (0.4, 0.9)$ on two NVIDIA RTX 3090 GPUs with 24 GiB RAM. Training takes four days for images of 288p resolution. Higher resolution training at 576p is only possible with a batch size of six and takes up to three weeks.

For 288p images, we train the networks for 300k-350k iterations and for 576p images we increase the number of iterations accordingly to adjust to the smaller batch size. Within this range, we use early stopping based on the validation score to select the model for evaluation. Furthermore, we train all our models with a learning rate of 1e-4, and a final tangent hyperbolic (tanh) non-linearity unless we explicitly specify it with "lin" (linear).

4. Evaluation

We evaluate our method on the challenging RealEstate10K dataset introduced by Zhou et al. [48], which contains approx. 80k video clips extracted from real estate YouTube videos, showing mostly indoor scenes. The videos have been automatically annotated with camera intrinsics and camera trajectories using ORB-SLAM2 [20] and bundle adjustment. Monocular SLAM cannot recover global scale, so the sequences have been scaled so that the near geometry lies at approx. 1.25 m [48].

Since some of the YouTube videos cannot be downloaded anymore, we only managed to obtain 74.5k video clips. We split the official *train* split further into 54k training and 13.5k validation sequences. All our tests are done using the official test split, except the extrapolation experiments, where we use the data provided in [32].

Interpolation. The first interesting problem setting is interpolation, i.e. when the target camera pose is roughly between the two input frames. For this, we randomly choose a target image $\tilde{I}_O$ and source frames Δt before (I_1) and after (I_2) it. Δt is uniformly sampled from the interval $[4, 13]$. Note that extrapolation areas may still occur in this mode, since the camera never moves perfectly on a straight line. We train and evaluate with a resolution of 518×288 unless otherwise mentioned.

Extrapolation. In order to investigate whether FaDIV-Syn is also suitable for extrapolation, we add extrapolation triplets to the training. These are randomly sampled from the video sequences, ensuring a distance $d_1 \in [3, 5]$ between the input frames I_0, I_1, as well as that the target frame is $d_2 \in [5, 7]$ frames after I_2. Extrapolation and interpolation triplets are mixed 80:20 during training and training is started from a pre-trained interpolation network. Accordingly, we only train 200k-250k iterations.

Shih et al. [32] evaluated different state-of-the-art extrapolation approaches, with a resolution of 1024×576. For a fair comparison, we introduce a variation of our FaDIV network called FaDIV-Big with one more downsampling layer and a larger gated group convolution kernel size ($k_1 = 7$ and $k_2 = 5$). In our evaluation, we follow the setup of [32] to be comparable.

For both modes, we evaluate the quality of generated images with the PSNR, SSIM [49], and LPIPS [47] metrics.

4.1. (Gated) Group Convolutions

We compare a FaDIV network with gated group convolutions and group convolutions without gating. Furthermore, we implemented a variation that uses structural-similarity maps as input for all layer pairs, as described in Sec. 3.4. Note that all networks share the same base network (including gated convolutions). To save time, we perform this experiment on a 35% fraction of the real dataset and train only for 200k-250k iterations. As Table 1 shows, the gated variant give better results on the test dataset. Appending pre-estimated structural similarity maps in the input performs a little better for image triplets with further distances Δt. We conclude that similarity maps are helpful, but this effect is less pronounced for closer image triplets.

	$\Delta t = 2$			$\Delta t = 5$			$\Delta t = 10$		
Variant	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
no-GGC	35.358	.9626	.0182	32.087	.9388	.0334	28.408	.8923	.0631
GGC	**36.150**	.9658	**.0158**	32.523	.9417	**.0307**	28.594	.8944	**.0605**
GGC+SSIM	35.860	**.9661**	.0170	**32.541**	**.9436**	.0314	**28.748**	**.8984**	.0609

Table 1: Evaluation of gated group convolutions (GGCs).

4.2. Interpolation Results

We achieved the best interpolation results with a FaDIV-Net with tanh activation (see Table 2). Again, we noticed that the SSIM variant shows an increasingly better performance for further image distances (especially for LPIPS).

In order to push the boundaries of view interpolation inference speed, we investigate a very fast variation with only 13 depth layers and tanh activation (full-13). Comparing the results shows that FaDIV-full-13 still has very good performance but cannot match the results of a FaDIV network with 17 layers. Averaged across all metrics and image distances we obtain a performance decrease of only -3.79%.

4.3. Generalization and Data Efficiency

Data Efficiency. To investigate the dependency of FaDIV-Net on available training samples, we train FaDIV networks on smaller subsets (5% & 35%) of the training dataset. Interestingly, for 35%, full-17-lin achieves same or better performance than the one trained on the full dataset (see Table 9). Further, FaDIV-Net already achieves very good results with only the base-network architecture (Sec. 3.3). If we train on 5% of the data, we loose only -2.1% in performance compared to the full training set. We conclude that FaDIV-Net has very good generalization ability and thus already reaches similar level of performance with significantly fewer data.

	$\Delta t = 2$			$\Delta t = 5$			$\Delta t = 10$		
Variant	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
full-17-lin	36.120	.9669	.0159	32.418	.9420	.0321	28.526	.8940	.0620
full-17	**36.485**	**.9684**	.0155	**32.799**	**.9452**	.0304	28.820	.8983	.0606
full-17-ssim	36.024	.9662	**.0151**	32.662	.9450	**.0295**	**28.870**	**.9006**	**.0590**
full-13	34.848	.9635	.0172	31.666	.9376	.0326	28.056	.8881	.0634

Table 2: Interpolation results on RealEstate10k.

Train		$\Delta t = 2$			$\Delta t = 5$			$\Delta t = 10$		
	Model	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
5%	full-17-lin	35.187	.9616	.0173	31.919	.9366	.0327	28.196	0.8874	.0631
35%	base17-lin	35.847	.9655	.0161	32.271	.9403	.0314	28.406	.8918	.0615
	full-17-lin	**36.150**	.9658	**.0158**	**32.523**	.9417	**.0307**	**28.594**	**.8944**	**.0605**
100%	full-17-lin	36.120	**.9669**	.0159	32.418	**.9420**	.0321	28.526	.8940	.0620

Table 3: Data efficiency experiment. The *Train* column shows the training dataset size relative to the full RealEstate10k train split.

Generalization to higher resolutions. We also evaluate full-17 and the SSIM variant on higher-resolution images. The networks are trained for image sizes of 512×288 and are afterwards tested for image sizes of 1024×576. The comparison of the networks in Table 4 and Table 5 shows that FaDIV-Syn generalizes surprisingly well for higher resolution images. Furthermore, it seems that using structural similarity maps gives a positive contribution to maintaining quality. This high quality can also be seen in Fig. 9.

4.4. Extrapolation

Shih et al. [32] evaluated an array of related methods for extrapolation tasks. In order to test view synthesis accuracy, they generated 1500 random triplets from the test data set. Note that these experiments were carried out at 1024×576,

Size		$\Delta t = 2$			$\Delta t = 5$			$\Delta t = 10$		
	Variant	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
288p	full-17	**36.485**	**.9684**	.0155	**32.799**	**.9452**	.0304	28.820	.8983	.0606
	+SSIM	36.024	.9662	**.0151**	32.662	.9450	**.0295**	**28.870**	**.9006**	**.0590**
576p	full-17	**35.713**	**.9575**	.0236	31.303	.9254	.0496	26.993	.8677	.1017
	+SSIM	35.360	.9552	**.0230**	**31.315**	**.9265**	**.0484**	**27.134**	**.8728**	**.0993**

Table 4: Interpolation results for generalization at inference time to a higher resolution (576p). All models are trained with 288p and have 17 depth layers.

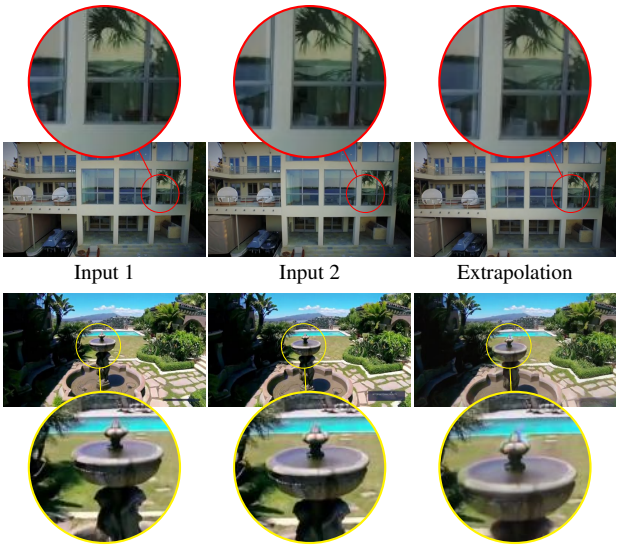


Figure 6: Extrapolation on the RealEstate10k test set.

double the resolution of our previous experiments. As Table 5 shows, we outperform current state-of-the-art methods on RealEstate10K. It is interesting that FaDIV-Syn learns to recognize inpainting areas even in disocclusion areas, as can be seen in Figs. 1 and 6.

Table 5 shows that all our models outperform state-of-the-art results in LPIPS and PSNR. Our SSIM results are better than [18, 36, 6], however Zhou et al. [48], Shih et al. [32] report better SSIM results. The last row of Table 5 shows our results on smaller images with 512×288, which is interesting but not directly comparable. For a direct comparison with Shih et al. [32], we refer to the appendices and Fig. 7. As shown in Table 5, FaDIV-Syn also generalizes to higher resolutions for extrapolation tasks. We outperform state-of-the-art results (LPIPS and PSNR) with a network that has never been trained on images with a resolution of 1024×576. By finetuning to 576p, we only improve the results slightly.

Method	576p	SSIM↑	PSNR↑	LPIPS↓
Stereo-Mag [48]	✓	**0.8906**	26.71	0.0826
PB-MPI (32 Layers) [36]	✓	0.8717	25.38	0.0925
PB-MPI (64 Layers) [36]	✓	0.8773	25.51	0.0902
PB-MPI (128 Layers) [36]	✓	0.8700	24.95	0.1030
LLFF [18]	✓	0.8062	23.17	0.1323
Xview [6]	✓	0.8628	24.75	0.0822
3D-Photo [32]	✓	0.8887	27.29	0.0724
FaDIV-17-BIG↓	✓	0.8800	**28.13**	**0.0634**
FaDIV-17	✓	0.8790	**28.13**	0.0674
FaDIV-17		0.8750	27.96	0.0695
(FaDIV-17 tested on 288p)		(0.8990)	(29.25)	(0.0426)

Table 5: Extrapolation results on RealEstate10k [48]. All methods without ✓are trained with 288p, but evaluated on 576p.

	Timings [ms]						Relative Metrics [%]		
	Torch		TRT-32		TRT-16		TRT-16		
Model	288p	540p	288p	540p	288p	540p	SSIM ↑	PSNR ↑	LPIPS ↓
full-13	12.4	41.0	9.8	30.8	4.2	15.0	100.03	99.99	99.90
full-17	15.9	52.0	11.3	35.8	4.8	18.3	99.98	99.99	100.04
base-17	12.6	42.0	9.5	29.8	3.3	11.8	100.00	99.99	99.92
big-17	24.5	78.0	19.5	62.9	8.0	26.2	99.98	99.99	100.03

Table 6: Inference times (ms) of generator networks on RTX 3090. We show native PyTorch as well as TensorRT (TRT) [22] float32/float16 versions. We also analyze the quantized model quality relative to the float32 models (relative metrics). The times do not include PSV generation (1.5 ms @ 540p).

The largest errors mainly occur in scenes with landscapes where objects are very far away from the camera. Unlike [48, 36] we use significantly fewer depth planes, where the last plane is at a distance of 16 m. Therefore, it is very challenging for FaDIV-Net to correctly generate the target view for scenes with large distances. We conclude that a larger depth range or techniques such as depth plane resampling [21] would probably help in these cases.

4.5. Inference Time

Table 6 shows the inference time of the different models on one NVIDIA RTX 3090 GPU. The fastest model is FaDIV-full-13, as it only uses 13 depth layers. FaDIV-BIG was designed to be comparable with non-real-time approaches. Nevertheless, it is still fast on large images, as we mainly enhanced the receptive field in the group convolution layers.

In addition to results on vanilla PyTorch, Table 6 also shows inference times in TensorRT [22], which already boosts performance for float32 precision without losing accuracy. We achieve a throughput of approx. 30 fps for our full models on 540p. TensorRT offers possibilities to quantize the weights of neural networks to float16 or even int8. Our test shows that float16 quantization retains almost 100% accuracy and leads to a significant performance boost: We can achieve up to 85 fps on 960×540 images and more than 300 fps for 512×288 resolution. In comparison, the approaches beating our SSIM score, 3D-Photo and Stereo-Mag, take 2-3 min and 93 ms per 540p image.

4.6. Qualitative Results

Continuous Depth. The fixed depth planes do not constrain FaDIV-Syn. This effect can especially be seen on straight lines across different depths, which are preserved by our approach (see Fig. 1 and supplementary material). We conclude that FaDIV-Net does not directly propagate information from the warped layers, but uses them as an orientation.

Occlusions & Disocclusions. Figs. 1 and 6 show examples of disocclusions. FaDIV-Syn can handle disocclusions

Figure 7: Extrapolation results of our method (left) compared with ground truth (center) and Shih et al. [32] (right).

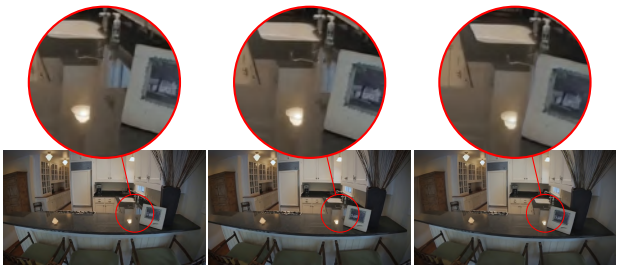


Figure 8: FaDIV-Syn is able to represent multiple layers of depth at one location and thus handles reflections correctly.

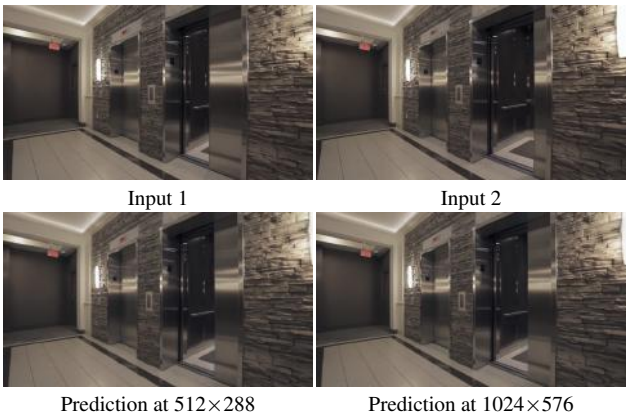


Figure 9: Interpolation from two reference views. The lower right video shows the results of a full-17-ssim network trained on 288p but evaluated on 576p. This figure is animated—if the video does not play, we refer to our supplementary material.

in both interpolation and extrapolation tasks. FaDIV-Syn separates the pixel information into different depth layers, detects correspondences in these and fuses the layers together. As Figs. 1 and 8 show, FaDIV can handle and represent occlusions.

Reflections and Transparency. Often methods that use depth have problems representing transparencies and reflections, since there is often more than one depth value at a certain pixel location. FaDIV-Syn is designed in such a way that there is not only one depth for each pixel, but a multitude of information in the different depth layers. This allows recognition of the correct position of both the surface and the reflection on it, as shown in Fig. 8. Examples for transparencies can be found in Figs. 1 and 7.

Moving Objects. The RealEstate10k dataset does not only contain static scenes. While dynamic scenes are more difficult for view synthesis (and indeed, one could argue that the problem is ill-posed), FaDIV-Syn nonetheless shows interesting behavior on such scenes. For example, the network has learned to interpolate between different positions of movable objects (see Fig. 9). This behavior would be hard to achieve using a pipeline which estimates depth first, or only blends pixel information.

4.7. Limitations

Our approach sometimes results in blurred objects under large camera movements. One example is the fountain shown in Fig. 6. We believe that one reason for this is that we have only trained with smaller camera offsets for extrapolation. Furthermore, we expect that semi-supervised techniques such as [10] could be used to increase the robustness of the method against arbitrary target poses, since the supervision offered by RealEstate10k only covers inter- and extrapolation on smooth camera trajectories. Additionally, the supported depth range is currently limited and could be extended as discussed in Sec. 4.4. Finally, the inference time is limited by the network itself, where compression techniques [16] could be applied to reduce network runtime even further.

5. Conclusion

We introduced FaDIV-Syn, a fast depth-independent novel view synthesis method. We demonstrated state-of-the-art performance in extrapolation on the RealEstate10k dataset. The method generalizes well to larger resolutions. Furthermore, our method is real-time-capable with 85-300 fps depending on output resolution. We think direct usage of the PSV for RGB view synthesis is a promising approach especially for real-time applications and further research in this direction is warranted.

Acknowledgement. This work was funded by grant BE 2556/16-2 of the German Research Foundation (DFG).

References

[1] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky. Neural point-based graphics. 2020. 2

[2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin. MatryODShka: Real-time 6DoF video view synthesis using multi-sphere images. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[3] R. O. Cayon, A. Djelouah, and G. Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *2015 International Conference on 3D Vision*, pages 469–477, 2015. doi: 10.1109/3DV.2015.59. 2

[4] X. Chen, J. Song, and O. Hilliges. Monocular neural image based rendering with continuous view control. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[5] S. Cheng, Z. Xu, S. Zhu, Z. Li, L. E. Li, R. Ramamoorthi, and H. Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2521–2531, 2020. 1

[6] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. In *IEEE International Conference on Computer Vision (CVPR)*, pages 7781–7790, 2019. 2, 7

[7] R. T. Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–363, 1996. 1, 3

[8] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941, 2017. 2, 5

[9] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner. Dynamic neural radiance fields for monocular 4D facial avatar reconstruction, 2020. arXiv preprint 2012.03065. 3

[10] N. Hani, S. Engin, J.-J. Chao, and V. Isler. Continuous object representation networks: Novel view synthesis without target view supervision. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 8

[11] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016. 2

[12] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 37(6):257:1–257:15, 2018. 2

[13] P. Huang, K. Matzen, J. Kopf, N. Ahuja, and J. Huang. DeepMVS: Learning multi-view stereopsis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2821–2830, 2018. 1

[14] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon. Dpsnet: End-to-end deep plane sweep stereo. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 1

[15] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 35(6), 2016. 1, 2

[16] J.-H. Luo, J. Wu, and W. Lin. ThiNet: A filter level pruning method for deep neural network compression. In *International Conference on Computer Vision (ICCV)*, 2017. 8

[17] D. C. Luvizon, G. S. P. Carvalho, A. A. dos Santos, J. S. Conceicao, J. L. Flores-Campana, L. G. L. Decker, M. R. Souza, H. Pedrini, A. Joia, and O. A. B. Penatti. Adaptive multiplane image generation from a single internet picture, 2020. 2

[18] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2, 7

[19] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 3

[20] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 5

[21] P. Nguyen, A. Karnewar, L. Huynh, E. Rahtu, J. Matas, and J. Heikkila. RGBD-Net: Predicting color and depth images for novel views synthesis. *arXiv preprint arXiv:2011.14398*, 2020. 2, 7

[22] NVIDIA. TensorRT. https://developer.nvidia.com/tensorrt. 7

[23] K. Olszewski, S. Tulyakov, O. Woodford, H. Li, and L. Luo. Transformable bottleneck networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7647–7656, 2019. 2

[24] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. Goldman, S. Seitz, and R. Martin-Brualla. Deformable neural radiance fields, 2020. arXiv preprint: 2011.12948. 3

[25] E. Penner and L. Zhang. Soft 3d reconstruction for view synthesis. 36(6), 2017. 2

[26] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. 2020. arXiv preprint: 2011.13961. 3

[27] G. Riegler and V. Koltun. Free view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 623–640. Springer, 2020. 2

[28] G. Riegler and V. Koltun. Stable view synthesis. *arXiv preprint arXiv:2011.07233*, 2020. 2

[29] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28. 4

[30] B. Ruf, B. Erdnüß, and M. Weinmann. Determining plane-sweep sampling points in image space using the cross-ratio for image-based depth estimation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W6:325–332, 08 2017. 1

[31] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 231–242, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919998. 3

[32] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3D photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 5, 6, 7, 8, 12, 13, 14, 15

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 2014. 5

[34] N. Snavely, R. Tucker, and S. Tulsiani. Layer-structured 3D scene inference via view synthesis. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[35] P. P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, and R. Ng. Learning to synthesize a 4D RGBD light field from a single image. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[36] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–184, 2019. 1, 2, 4, 7

[37] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library, 2020. 2

[38] J. Thies, M. Zollhöfer, C. Theobalt, M. Stamminger, and M. Nießner. Image-guided neural object rendering. In *International Conference on Learning Representations*, 2020. 2, 4

[39] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video, 2020. arXiv preprint 2012.12247. 3

[40] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[41] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser. Ibrnet: Learning multi-view image-based rendering. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3

[42] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. SynSin: End-to-end view synthesis from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[43] J. Xie, R. Girshick, and A. Farhadi. Deep3D: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, volume 9908, pages 842–857. Springer, 10 2016. 2

[44] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelNeRF: Neural radiance fields from one or few images. *arXiv preprint arXiv:2012.02190*, 2020. 3

[45] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 5

[46] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 3

[47] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[48] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 1, 2, 4, 5, 7

[49] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. 5, 6

[50] C. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23:600–608, 2004. doi: 10.1145/1015706.1015766. 3

Appendices

A. Network Architecture Details

We provide more detail regarding the network architecture in Tables 7 and 8.

Input	k_1	c_1	k_2	c_2	Output
P	3	102	3	204	$groupconv$
$\mathbf{G}(groupconv)$	3	112	3	112	$bottleneck$
$\mathbf{G}(bottleneck)$	3	112	3	224	$down_1$
$\mathbf{G}(down_1)$	3	128	3	256	$down_2$
$\mathbf{G}(down_2)$	3	256	3	512	$down_3$
$\mathbf{G}(down_3)$	3	256	3	1024	$down_4$
$\mathbf{G}(down_4)$	3	512	3	512	$dilated$
$\mathbf{Up}(dilated, G(down_3))$	3	512	3	256	up_1
$\mathbf{Up}(up_1, G(down_2))$	3	256	3	224	up_2
$\mathbf{Up}(up_2, G(down_1))$	3	224	3	112	up_3
$\mathbf{Up}(up_3, G(bottleneck))$	3	112	3	32	up_4
up_4	1	3	–	–	$pred$

Table 7: FaDIV-Full-17 architecture. Each row shows 2 convolutional layers, where k is the kernel size and c is the number of output features. **G** denotes the gating operation and **Up** is a transposed convolution for upsampling.

Input	k_1	c_1	k_2	c_2	Output
P	7	102	5	204	$groupconv$
$\mathbf{G}(groupconv)$	3	112	3	112	$bottleneck$
$\mathbf{G}(bottleneck)$	3	112	3	224	$down_1$
$\mathbf{G}(down_1)$	3	128	3	256	$down_2$
$\mathbf{G}(down_2)$	3	256	3	512	$down_3$
$\mathbf{G}(down_3)$	3	256	3	1024	$down_4$
$\mathbf{G}(down_4)$	3	512	3	2048	$down_5$
$\mathbf{G}(down_5)$	3	1024	3	1024	$dilated$
$\mathbf{Up}(dilated, G(down_4))$	3	1024	3	512	up_1
$\mathbf{Up}(up_1, G(down_3))$	3	512	3	256	up_2
$\mathbf{Up}(up_2, G(down_2))$	3	256	3	224	up_3
$\mathbf{Up}(up_3, G(down_1))$	3	224	3	112	up_4
$\mathbf{Up}(up_4, G(bottleneck))$	3	112	3	64	up_5
up_5	1	3	–	–	$pred$

Table 8: FaDIV-Big-17 architecture. Each row shows 2 convolutional layers, where k is the kernel size and c is the number of output features. The second block ($bottleneck$) contains one convolution (k_1,c_1) and two convolutions (k_2, c_2).

B. Extended Data Efficiency Results

We present more details on the data efficiency experiments, which where omitted due to space limitations from Section 4.3. As explained there, we train on smaller fractions of the full RealEstate10k training dataset, and evaluate on the full test set. The dataset size is reduced by randomly choosing scenes until the specified size is met (35%, 5%, 1%, and 0.1%). As Table 9 shows, all sizes from 35% to 1% give sufficiently good results, where 35% even performs similarly or slightly better than our model trained on the full dataset. It is possible that further training may yield advantages, since we set an upper bound on the training iterations as described in Section 3.5. However, we conclude that 35% of RealEstate10k still contains enough scene and pose variance to prevent the network from overfitting (see Fig. 13). This is to be expected, since the triplet sampling during training greatly augments the number of training samples. The 1% network maintains good performance for SSIM and PSNR but starts losing significantly in LPIPS. Finally, the 0.1% network loses significant performance in all metrics and seems to be outside of the boundary for satisfactory results. As Figs. 10 and 11 show, we observed significant

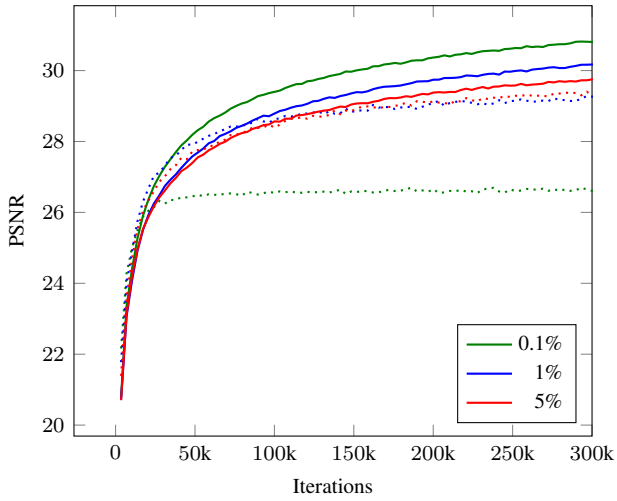


Figure 10: PSNR on reduced training (solid) and validation (dotted) splits of the RealEstate10k dataset during generator network training.

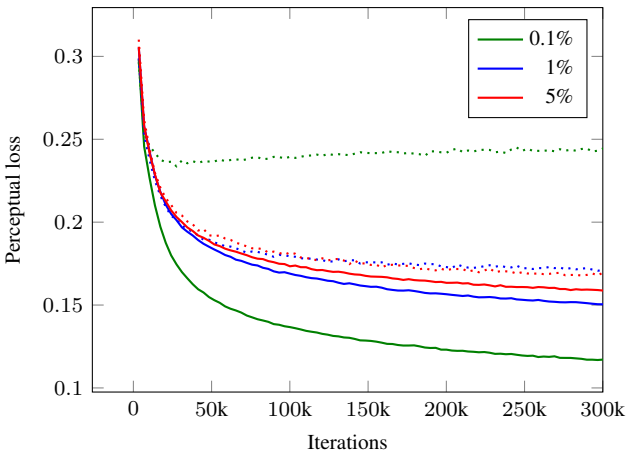


Figure 11: Perceptual loss on reduced training (solid) and validation (dotted) splits of the RealEstate10k dataset during generator network training.

Train	Model	$\Delta t = 2$			$\Delta t = 5$			$\Delta t = 10$		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
0.1%	full-17-lin	$31.91_{-11.7\%}$	$.9361_{-3.19\%}$	$.0381_{+140\%}$	$28.03_{-13.5\%}$	$.8825_{-6.32\%}$	$.0712_{+122\%}$	$24.70_{-13.4\%}$	$.8129_{-9.07\%}$	$.1257_{+103\%}$
1%	full-17-lin	$34.93_{-3.31\%}$	$.9607_{-0.64\%}$	$.0191_{+20.1\%}$	$31.38_{-3.20\%}$	$.9320_{-1.06\%}$	$.0360_{+12.2\%}$	$27.60_{-3.24\%}$	$.8789_{-1.69\%}$	$.0695_{+12.1\%}$
5%	full-17-lin	$35.19_{-2.58\%}$	$.9616_{-0.55\%}$	$.0173_{+8.81\%}$	$31.92_{-1.54\%}$	$.9366_{-0.57\%}$	$.0327_{+1.87\%}$	$28.20_{-1.16\%}$	$.8874_{-0.74\%}$	$.0631_{+1.77\%}$
35%	base17-lin	$35.85_{-0.76\%}$	$.9655_{-0.15\%}$	$.0161_{+1.26\%}$	$32.27_{-0.45\%}$	$.9403_{-0.18\%}$	$.0314_{-2.18\%}$	$28.41_{-0.42\%}$	$.8918_{-0.25\%}$	$.0615_{-0.81\%}$
	full-17-lin	$\mathbf{36.15}_{+0.08\%}$	$.9658_{-0.11\%}$	$\mathbf{.0158}_{-0.63\%}$	$\mathbf{32.52}_{+0.32\%}$	$.9417_{-0.03\%}$	$\mathbf{.0307}_{-4.36\%}$	$\mathbf{28.59}_{+0.24\%}$	$\mathbf{.8944}_{+0.05\%}$	$\mathbf{.0605}_{-2.42\%}$
100%	full-17-lin	$36.12_{+0.00\%}$	$\mathbf{.9669}_{+0.00\%}$	$.0159_{+0.00\%}$	$32.42_{+0.00\%}$	$\mathbf{.9420}_{+0.00\%}$	$.0321_{+0.00\%}$	$28.53_{+0.00\%}$	$.8940_{+0.00\%}$	$.0620_{+0.00\%}$

Table 9: Data efficiency experiment. The *Train* column shows the training dataset size relative to the full RealEstate10k train split.

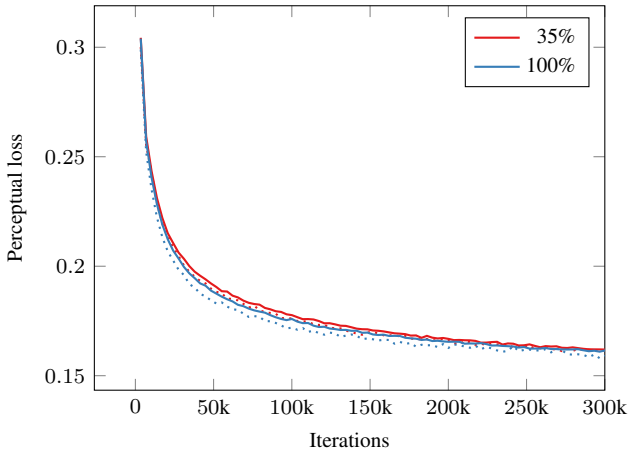


Figure 12: Perceptual loss on reduced training (solid) and validation (dotted) splits of the RealEstate10k dataset during generator network training.

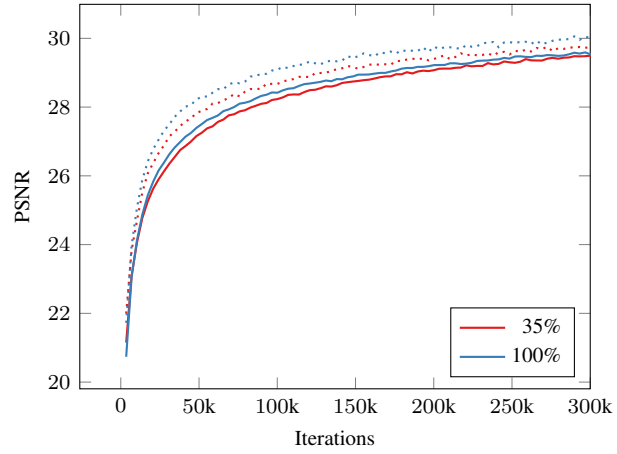


Figure 13: PSNR on reduced training (solid) and validation (dotted) splits of the RealEstate10k dataset during generator network training.

drops in validation performance for the 1% and 0.1% training split. Starting with 35%, we observe that the validation score is actually better than the training score (see Figs. 12 and 13), which is caused by batch normalization: The average parameters used during evaluation seem to work more robustly than the on-line statistics computed for each batch during training. Overall, we conclude that above 35% there are no indications of overfitting at all.

C. Additional Qualitative Results

In addition to the exemplary results already shown, we present more qualitative examples here. Figures 14 and 15 show extrapolation examples for our BIG and Full network variants, in comparison to ground truth and 3D Photo [32]. To further demonstrate the generalization capability of our method to higher resolutions, we compare networks trained in 288p and 540p in Fig. 16. Finally, we show interpolation sequences in Figs. 17 and 18.

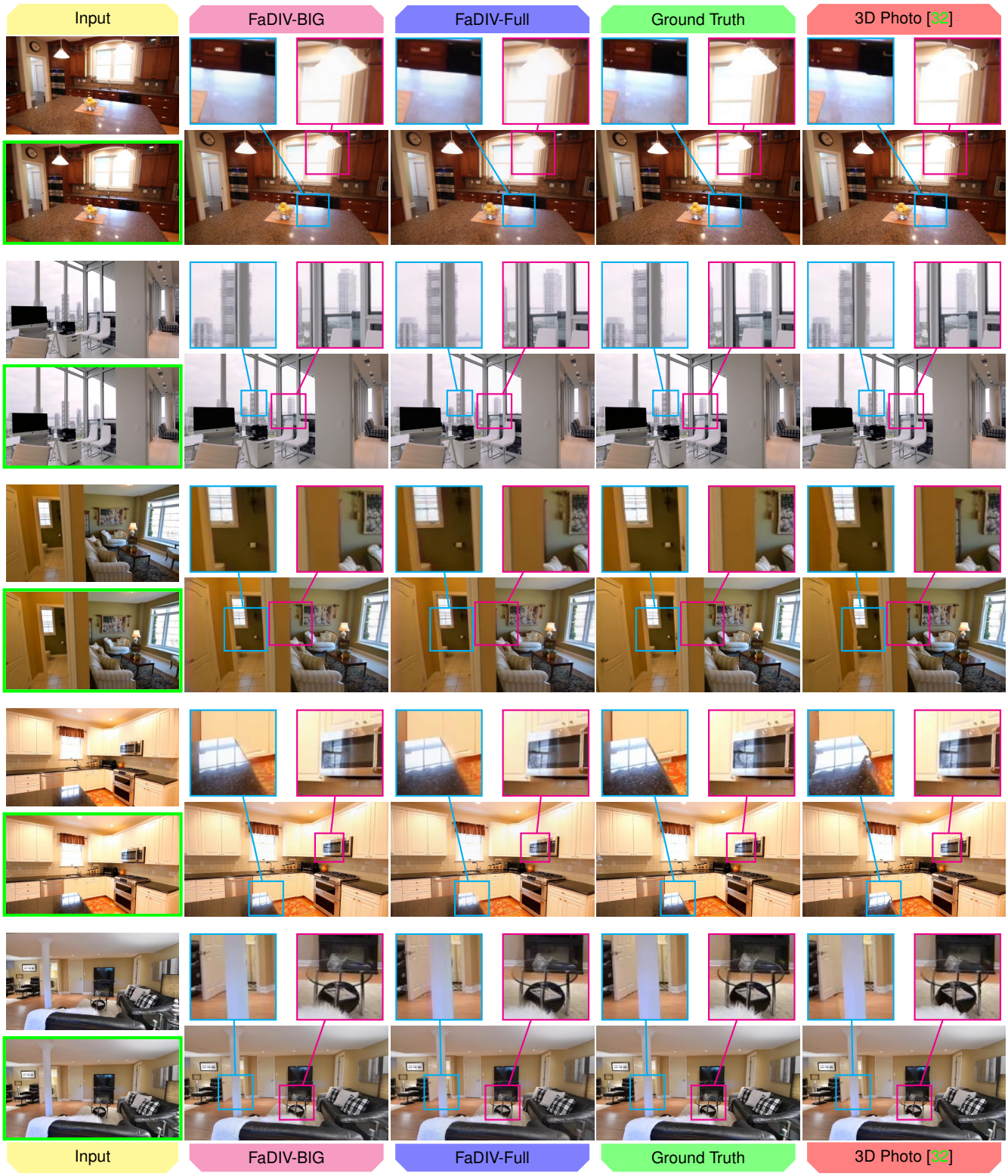


Figure 14: Extrapolation comparison of our FaDIV-17-BIG and FaDIV-Full-17 (fast) networks against ground truth and 3D Photo [32]. The input frame closer to the target frame is marked in green for easier comparison.

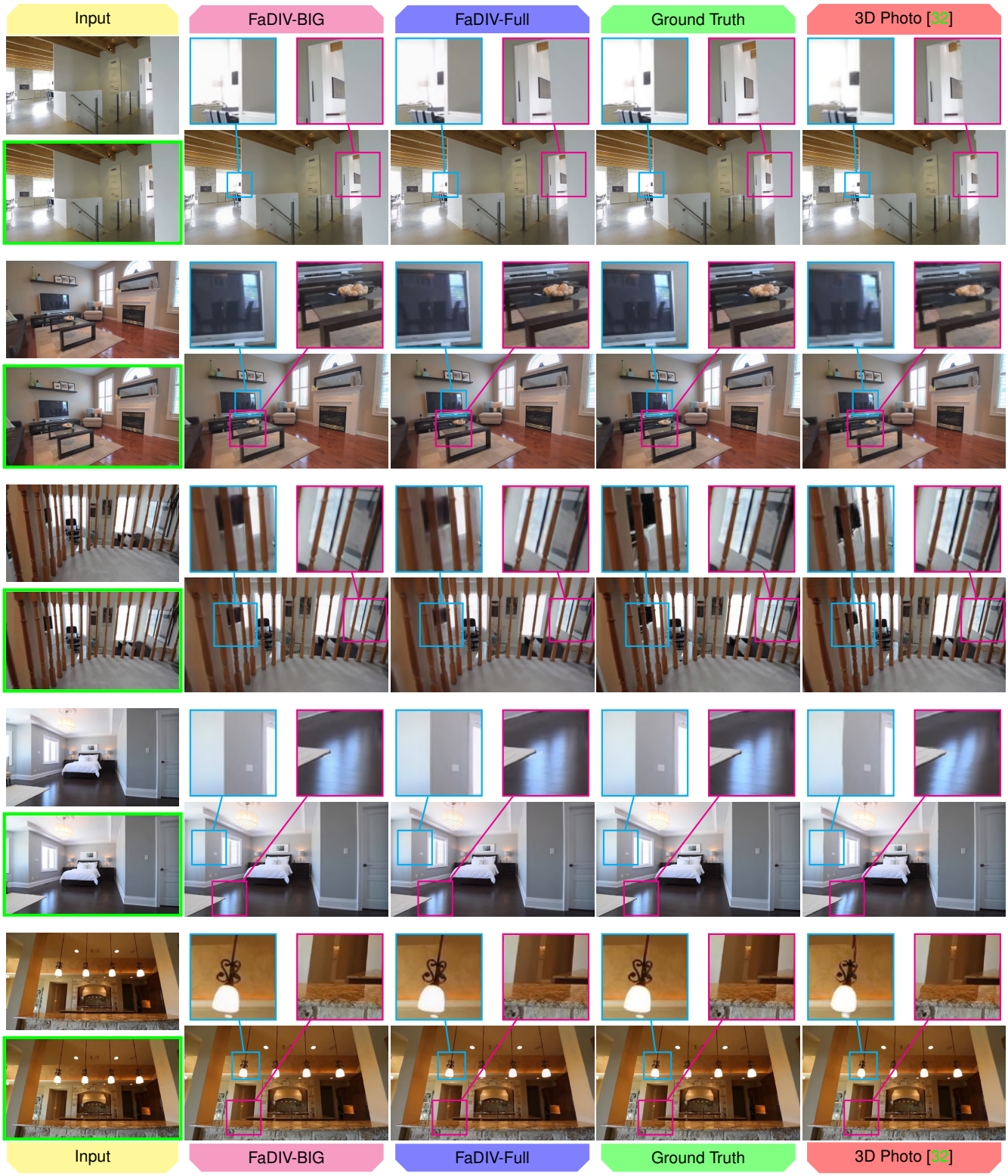


Figure 15: Extrapolation comparison of our FaDIV-17-BIG and FaDIV-Full-17 (fast) networks against ground truth and 3D Photo [32]. The input frame closer to the target frame is marked in green for easier comparison.

Figure 16: Extrapolation comparison of FaDIV-Full-17 trained on 540p (Full) and 288p (Full-17-Generalized) against ground truth and 3D Photo [32]. The Full-17-Generalized network runs inference in 540p. The input frame closer to the target frame is marked in green for easier comparison.

Figure 17: Interpolation using our fast FaDIV-Full-17 network. In every block, the top row (green) shows the ground truth trajectory from RealEstate10k, while the bottom row (blue) presents the interpolated result corresponding to the ground truth camera poses.

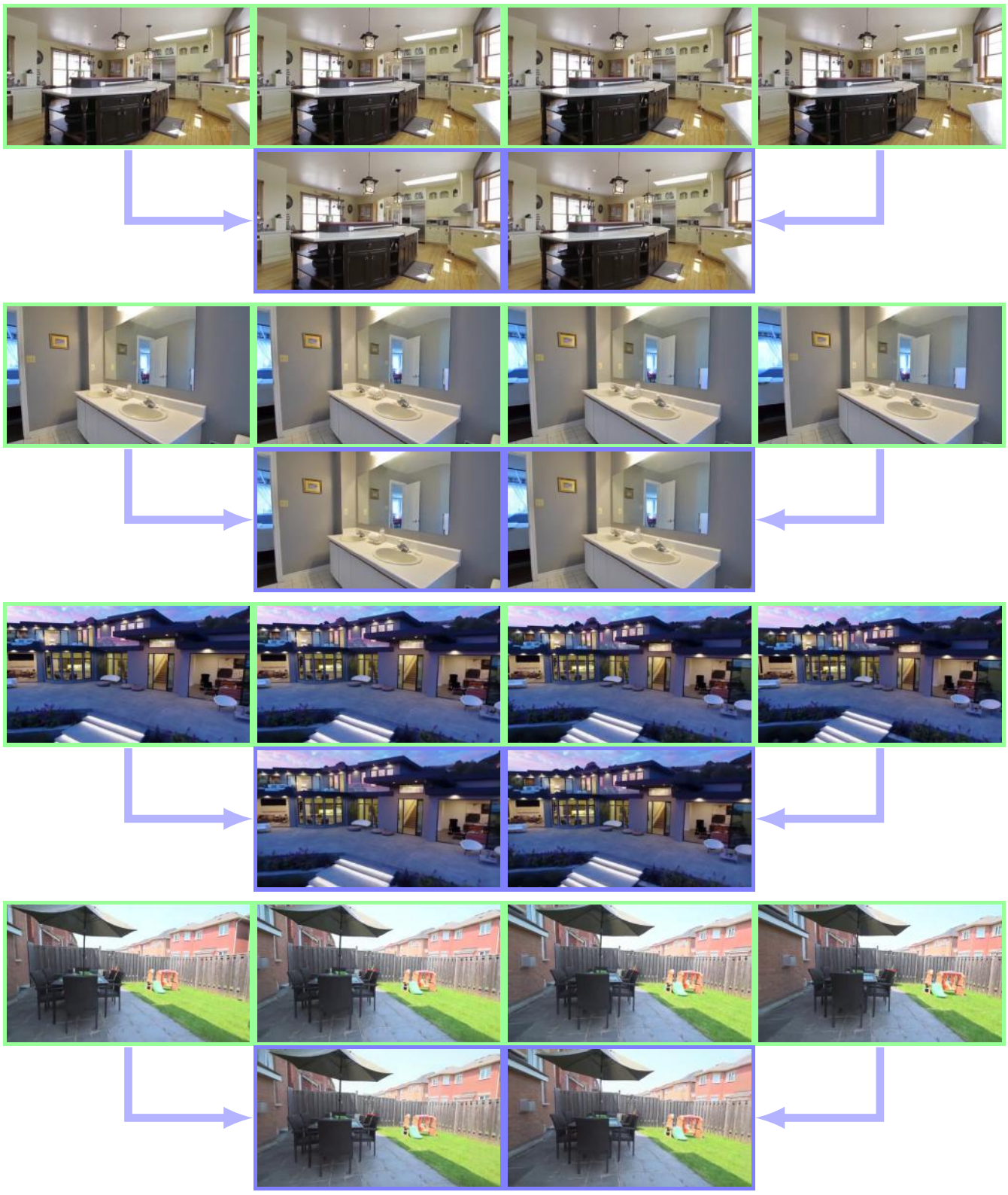


Figure 18: Interpolation using our fast FaDIV-Full-17 network. In every block, the top row (green) shows the ground truth trajectory from RealEstate10k, while the bottom row (blue) presents the interpolated result corresponding to the ground truth camera poses.