FaDIV-Syn: Fast Depth-Independent View Synthesis using Soft Masks and Implicit Blending

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

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

Michael Weinmann  
Delft University of Technology

Sven Behnke  
University of Bonn

Abstract—Novel view synthesis is required in many robotic applications, such as VR teleoperation and scene reconstruction. Existing methods are often too slow for these contexts, cannot handle dynamic scenes, and are limited by their explicit depth estimation stage, where incorrect depth predictions can lead to large projection errors. Our proposed method runs in real time on live streaming data and avoids explicit depth estimation by efficiently warping input images into the target frame for a range of assumed depth planes. The resulting plane sweep volume (PSV) is directly fed into our network, which first estimates soft PSV masks in a self-supervised manner, and then directly produces the novel output view. This improves efficiency and performance on transparent, reflective, thin, and feature-less scene parts. FaDIV-Syn can perform both interpolation and extrapolation tasks at 540p in real-time and outperforms state-of-the-art extrapolation methods on the large-scale RealEstate10k dataset. We thoroughly evaluate ablations, such as removing the Soft-Masking network, training from fewer examples as well as generalization to higher resolutions and stronger depth discretization. Our implementation is available.

I. INTRODUCTION

Novel view synthesis (NVS) aims to estimate images from novel viewpoints of a scene from images captured from one or more reference views.

This is of great relevance for numerous applications in virtual reality, 3D movies, computer games, and other areas where images have to be generated efficiently under arbitrarily chosen viewpoints. VR teleoperation of robots is a specific application (see Fig. 1). Here NVS can be employed to reduce latencies (e.g., instantaneously reacting to the viewpoint changes by the operator), to compensate camera workspace limits, or to generate third-person views. However, this application places special constraints on the NVS algorithm: Movement latency needs to be minimal to avoid VR sickness, i.e. novel views have to be generated in real time. On the other hand, scene latency, i.e. the time from image capture to display also needs to be minimal to allow efficient telemanipulation. This effectively rules out approaches which have a costly preprocessing step that generates a suitable representation for later interpolation. Note that NVS for VR teleoperation is of high importance for recently emerging Avatar systems [1, 2].

However, synthesizing novel views of a sparsely captured scene is challenging, as scene geometry and surface properties are unknown a priori and have to be inferred from the input 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.

Our work is focused 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 mounted on a robotic system. We note, though, that we also show applicability to more input views.

Many stereo [4–7] and view synthesis [3, 8, 9] methods use Plane Sweep Volumes (PSVs) [10], which warp the input views on a range of parallel planes 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 [9] or to generate representations such as multi-plane images [3, 8]. This approach exhibits a fundamental bottleneck, however: Imprecise or wrong depth estimates, which occur especially on uniform, transparent, thin, 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 (IBR) approaches, but forgoes the geometry estimation step by directly operating on the PSV in the target view to produce
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 convolutions and gated convolutions [11, 12], which are suitable for detecting layer-wise correspondences, masking of irrelevant areas within the layers, and blending. Unlike most IBR approaches, FaDIV-Syn has no explicit blending or inpainting stage, which avoids early and hard decisions, allowing the distribution of the blending and inpainting operations throughout the learned network. Only a single forward pass is required for view interpolation and extrapolation. Further, we demonstrate how gated convolutions can be used to learn self-supervised soft masks, which are predicted by a lightweight CNN to give a soft correspondence measure, which improves accuracy and provides insight on the network’s method of operation.

In summary, our contributions include 1) a real-time view synthesis method operating on plane sweep volumes without requiring explicit geometry, 2) self-supervised learning of soft-masks by introducing a gating module, and 3) a detailed evaluation on the large-scale RealEstate10k dataset [3], where we demonstrate our approach to outperform existing methods in terms of accuracy and runtime.

II. RELATED WORK

Novel view synthesis has gained much attention in recent years and a large variety of approaches has been presented. For a broader review, we refer to surveys by Nguyen et al. [13] and Tewari et al. [14].

Image Based Rendering (IBR): In contrast to classical rendering of 3D scenes using textured geometry, IBR methods render novel views by combining input images in the target pose [15–19, 9, 13, 20–24]. 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 [15–17, 22]. Hedman et al. [16] learn the blending operation end-to-end. Riegler and Kolten [22] use a recurrent blending decoder in order to deal with a varying number of input images. In recent work [23], heuristic input image selection is replaced with a fully-differentiable synthesis block. Penner and Zhang [15] introduce soft visibility volumes, which encode occlusion probabilities and thus avoid early decisions, but require a larger number of input views. Kalantari et al. [9] 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. [13] 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. Similar to our approach, Flynn et al. [24] also directly build a PSV in the target view. Color is fused for each PSV plane pair and a separate network estimates depth probabilities for blending. 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 geometry-based approaches [25–28] use depth features to spatially project pixel information and refine these projections to a target view. Wiles et al. [27] and Chen et al. [29] process single input images, estimating monocular depth in an end-to-end fashion. Wiles et al. [27] implement a differentiable point cloud renderer that allows z-buffering and splatting. Chen et al. [29] predict depth in the target view using a transforming auto-encoder that explicitly transforms the latent code before entering the decoder. Similarly, Olszewski et al. [30] learn implicit voxel representations and transform encoded representations explicitly. Srinivasan et al. [28] 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. Choi et al. [26] predict depth probabilities along camera rays in multiple input images and unite 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 [31] 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 a higher computational effort—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 [32, 33], Tucker and Snavely [33] train a network to estimate scale-invariant depth and require additional sparse point clouds to recover scale. Multiview approaches [8, 3, 34] use information from additional camera poses to place surfaces at the correct MPI layer. Plane sweeping [3] or warping [8] at different depths creates a suitable representation for the network. Mildenhall et al. [34] blend the layers of multiple MPIs to generate novel views with local light fields. Recently, Attal et al. [35] extended the key idea of multiplane images to multisphere images in order to synthesize 360° images in real-time, however, at lower resolution.

Other approaches [36–38] build on Layered Depth Images (LDI) [39], which store multiple RGB and depth values per pixel. Snavely et al. [36] train a CNN to predict a two-layered LDI, where the network learns to predict occluded pixels. Shih et al. [37] 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. It is applicable for dynamic scenes in real-time.
Neural Rendering: In recent years, view synthesis approaches based on Neural Radiance Fields (NeRF) [40] 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 [41, 42, 19, 43] show impressive results on a variety of scenes and recent developments also achieve real-time inference [44–47]. However, with the exception of Wang et al. [19] and Sitzmann et al. [47], NeRFs typically have to be trained with hundreds of images of the target scene, making them unsuitable for dynamic scenes. While methods designed for dynamic scenes exist [43, 48–50], they require offline training or processing phases as well.

III. PROPOSED METHOD

The key idea of FaDIV-Syn, illustrated in Fig. 2, is to preprocess and transform the input images into the target frame, without making early and hard decisions. Of course, this transformation requires depth information. Instead of explicitly estimating depth, which exhibits problems for transparent, reflective, thin, and featureless surfaces, we sample multiple depth values and present the resulting possibilities to a deep neural network for learning image synthesis. The induced multi-plane representation is well-suited for view synthesis.

We present and evaluate our method for two input images (in the following denoted by \( I_1 \) and \( I_2 \)) and one output image \( I_O \). This scenario commonly occurs in many contexts, such as robotics, AR/VR, and mobile devices. We note though, that additional input views enhance the performance further (see Sec. IV-B). For all 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 \( N \rightarrow \infty \), determine each pixel’s 3D position by searching for correspondences in the warped planes (see Figs. 2 and 4). When performing the synthesis 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.

A. Plane Sweep Volumes for View Synthesis

Planar geometry is especially well-suited for camera-to-camera projection, since the resulting warping operation can be performed efficiently. 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 [10]. For each plane \( i \) with depth \( d_i \), we define \( P_k^{(i)} \) as the image resulting from projecting \( I_k \) onto the plane, and then into \( I_O \). Using this representation, we can define a “hard-wired” view synthesis method \( f \):

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

\[
D(p) = \arg \max _j Q(P_{1}^{(j)}, P_{2}^{(j)}, p),
\]

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 be compared. Figure 4 shows an exemplary PSV where the idea presented in Eq. (1) is immediately apparent. Since perfect \( Q \) and \( g \) are, however, not known, we will learn a CNN approximating \( f \).

Depth Discretization: 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 most efficient network uses 19 depth planes, which is much less compared to related methods [8, 3] but achieves significantly better accuracy (see Sec. IV). The depths are sampled in disparity space, where we linearly interpolate within a chosen disparity range.

B. View Synthesis Network

When generating a novel view directly from a PSV, disocclusion areas are not trivial to determine. Thus, our network must learn to distinguish between (1) areas of sufficient correspondences in the warped planes, (2) areas of no correspondence but sufficient correspondence in different planes, 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 planes and recognize inpainting areas to fill them with realistic content. In contrast to Thies et al. [21], who also learn implicit blending, we avoid early and hard decisions in a

Fig. 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 approximate object depth. (b) The resulting projected images are stacked and fed into the view synthesis network (c), which directly predicts the target image.
Fig. 3: View extrapolation on RealEstate10k [3] test set. Please see our supplementary video for an animated version of this figure that shows both inter- and extrapolation along a trajectory.

Fig. 4: Plane sweep volume (PSV). Four planes of the scene in Fig. 3 with α-blended projections of the two input images. Note that RealEstate10k only provides estimated global scale, so the given plane distances are up to a scale. Exemplary areas with good correspondence between the two input views are marked in red. Forwarding the corresponding regions would already yield an approximate solution.

depth estimation stage, which may lead to quality degradation later in the pipeline.

Our network is divided into several components (see Fig. 5): A soft-masking network, a gating section, and the final fusion network.

1) Soft-Masking Network: The Soft-Masking (SM) network operates on the input PSV and generates approximate PSV masks that guide the further synthesis process. Ideally, the output mask \( m \) should be the correspondence quality measure \( Q(P_1^{(j)}, P_2^{(j)}) \) (see Eq. (1)). In practice, the entire pipeline is trained end-to-end without direct supervision on the masks. As shown in Fig. 6, the SM network learns meaningful correspondence masks that correlate closely with depth in the scene. Since there is a large amount of overlap, we conclude that the later fusion network uses the inferred masks mostly to eliminate areas of poor correspondence, but keeps multiple possibilities to make final decisions in deeper layers. We thus denote these representations as soft masks.

To generate \( m \), we implement an hourglass network module with four downsampling and four upsampling blocks. Each block has one convolution followed by average pooling or bilinear upsampling. The module processes the PSV \((P_1, P_2)\) as \(2 \times N\) input planes in grayscale and outputs \(2 \times N\) feature maps. For each plane, we encourage the network to learn a binary classification mask by applying the softmax function (separately for each plane). For details we refer to the supplementary material.

2) Learned Gating: We send the entire PSV through a gating section. This way, the network can eliminate areas of low correspondence.

Gated convolutions [11] have recently shown promising results in image inpainting [12]. Instead of \( C(W_f, I) = \sigma(W_f \ast I) \), a gating layer calculates the non-linear output

\[
GC(W_f, W_g, I) = \sigma_1(W_f \ast I) \odot \sigma_2(W_g \odot I),
\]

where \( \odot \) denotes the element-wise multiplication and \( \ast \) the convolution. \( W_g, W_f \) are two different convolutional filters, and \( \sigma_1, \sigma_2 \) are activation functions. The formula shows that gating directly influences the actual feature extraction and can thus adapt it to the context. Gating layers can help the network especially with performing masking-like operations (e.g. when recognizing corresponding depth planes), performing blending, or determining inpainting areas [12].

To avoid hard decisions, we allow the gating stage to incorporate the mask \( m \) in a learned fashion by adding convolutional layers that can modify the mask and features before the actual multiplication (see Fig. 5). We note that the network may also learn to ignore the provided mask.

Since the depth planes are aligned, we can be sure that corresponding areas only appear in corresponding planes (see Fig. 2). Hence, we can restrict the convolutions through grouping under the assumption that other planes are initially irrelevant to the considered plane pair.

We concatenate the estimated masks \( m \) to the input plane pairs \( P_1^{(j)}, P_2^{(j)} \) and activate the gating feature extraction with a sigmoid (\( \sigma \)), which creates a mask \( \tilde{m} \in [0, 1] \):
It is defined as $\Psi$ where the perceptual loss is based on a VGG-19 [52] network and intrinsics. We define the loss function following a $\tanh$ two for downsampling. The final output of the fusion network, and ReLU activation. The second convolution has a stride of block consists of two convolutions with batch normalization upsampling, we use bilinear interpolation. Each downsampling it is based on a U-Net [51] architecture and consists of the fusion and inpainting module (Fusion-Net in Fig. 5). early projection errors due to the depth discretization. complexity of the mask generator bounded, and (iii) avoid presentation and incorporation, we thus (i) keep the key idea (manner from a triple $[I_1, I_2, I_O]$ with known camera poses and intrinsics. We define the loss function

$$ L(\hat{I}_O, I_O) = \lambda_1 L_1(\hat{I}_O, I_O) + \lambda_2 L_{\text{perceptual}}(\hat{I}_O, I_O), \quad (7) $$

where the perceptual loss is based on a VGG-19 [52] network $\Psi$ pretrained on ImageNet. It is defined as

$$ L_{\text{perceptual}}(\hat{I}_O, I_O) = \sum_{l=1}^{L} \frac{w_l}{N_{\Psi_l}} |\Psi_l(\hat{I}_O) - \Psi_l(I_O)|, \quad (8) $$

where $\Psi_l(\cdot)$ is the activation of the $l$-th layer, $w_l$ is a weight factor of the $l$-th layer and $N_{\Psi_l}$ are the number of elements in the $l$-th layer. We train the network with a batch size of 20, a learning rate of 1e-4, and the Adam optimizer with $\beta_{1,2} = (0.4, 0.9)$ on two NVIDIA A6000 GPUs with 48 GiB RAM. Training takes four days for images of 288p resolution. Training for a higher resolution of 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, whereas 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.

## IV. Evaluation

We train and evaluate our method on the challenging RealEstate10k dataset introduced by Zhou et al. [3], which contains approx. 75k video clips extracted from YouTube videos, showing mostly indoor scenes. The large variety in the dataset allows generalizing to different scenarios (see Fig. 1). All videos have been automatically annotated with camera intrinsics and camera trajectories using ORB-SLAM2 [53] 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 [3]. We divide the official train split further into 54k training and 13.5k validation sequences. All our tests are done using the official test split, where the extrapolation experiments use the data provided by Shih et al. [37]. For all experiments, we evaluate the quality of generated images with the PSNR, SSIM [54], and LPIPS [55] metrics.

### A. Extrapolation

The first, more challenging task is extrapolation, i.e. the target view is outside of the provided input views. We start from a pre-trained interpolation network (see Sec. IV-B) and add extrapolation sequences. Extrapolation and interpolation triplets are mixed in the ratio 80% to 20% during training. These are randomly sampled from 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 $I_O$ is $d_2 \in [5, 7]$ frames after $I_1$.

Shih et al. [37] 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. We follow the same evaluation protocol. Note that these experiments are carried out at a resolution of $1024 \times 576$. 

$$ GGC_1^{(j)} = \text{ReLU}(W_1 \odot [P_1^{(j)}, P_2^{(j)}, \tilde{m}^{(j)})], \quad (4) $$

$$ \tilde{m}(j) = \sigma(W_2 \odot GGC_1^{(j)}, \quad (5) $$

$$ GGC_2^{(j)} = \text{ReLU}(W_3 \odot GGC_1^{(j)}) \odot \tilde{m}^{(j)}, \quad (6) $$

where $W$ denotes the learned weights.

By allowing self-supervised learning of a suitable mask presentation and incorporation, we thus (i) keep the key idea of our approach to not estimate depth explicitly, (ii) keep the complexity of the mask generator bounded, and (iii) avoid early projection errors due to the depth discretization.

3) Fusion Network: Finally, the gated PSV features enter the fusion and inpainting module (Fusion-Net in Fig. 5). Variant without soft masking (NoSM): To validate the effect of soft masking, we also tested generating novel views without the SM network. Here, the gating section receives only $P_1, P_2$ as input. In this setup, the network must learn to spread the correspondence estimation throughout the layers. We therefore use Gated Convolutions in each downsampling layer and increase the number of feature-maps in deeper layers. Note that both manipulations are parameter-intensive and increase the accumulated number of parameters by approximately 33%-55%, depending on the number of PSV planes.

Architecture details can be found in the supplementary.

## C. Training

The whole pipeline is trained end-to-end in a supervised manner from a triple $(I_1, I_2, I_O)$ with known camera poses and intrinsics. We define the loss function

$$ L(\hat{I}_O, I_O) = \lambda_1 L_1(\hat{I}_O, I_O) + \lambda_2 L_{\text{perceptual}}(\hat{I}_O, I_O), \quad (7) $$

$$ L_{\text{perceptual}}(\hat{I}_O, I_O) = \sum_{l=1}^{L} \frac{w_l}{N_{\Psi_l}} |\Psi_l(\hat{I}_O) - \Psi_l(I_O)|, \quad (8) $$

Fig. 6: Self-supervised soft masking. Predicted image (left) and learned mask activations (right). The PSV layer depth is up to scale. Note how the learned masks correlate with depth in the scene.
Results: As shown in Table I, we outperform current state-of-the-art methods on RealEstate10k by significant margins in all metrics. Reducing the number of depth layers decreases the performance only slightly (Ours-19). To evaluate the Soft-Masking network, we also evaluate ablations without it (NoSM). However, removing the Soft-Masking network results in a noticeable drop in performance. We note that NoSM variants still achieve significantly higher PSNR and LPIPS values than related methods (see Table I), however, they generally in all metrics. Reducing the number of depth layers clearly reduces performance. However, removing the soft-masking network (NoSM variant) has a much bigger impact, so that the F-19 network outperforms the F-32-NoSM variant with significantly more depth planes.

Three source views: In order to demonstrate FaDIV-Syn’s capability to handle more than two input views, we also report targeting image $I_O$ and source frames $I_1$ and $I_2$ before and after it, respectively. The distances $\Delta t_{1}, \Delta t_{2}$ are uniformly sampled from the interval $[4, 13]$. Note that extrapolation may still occur in this mode, since the camera seldomly moves perfectly on a straight line. We perform both training and evaluation based on the image resolution of $518 \times 288$ unless otherwise mentioned.

In Table II we compare our approach to our main competitor Stereo Magnification [3] for interpolation, which we outperform in all metrics. Note that our networks generalize from a training image size of $512 \times 288$ to $1024 \times 576$ at inference, while Stereo-Mag was trained for higher resolution explicitly. Looking at the ablations, we again achieve better generalization ability in the variants with soft masking (especially in comparison with Table III).

In our ablation study in Table III, reducing the number of PSV planes clearly reduces performance. However, removing the soft-masking network (NoSM variant) has a much bigger impact, so that the F-19 network outperforms the F-32-NoSM variant with significantly more depth planes.

B. Interpolation

The second interesting problem setting is interpolation, i.e. when the target camera pose is roughly between the two input frames of a video sequence. For this, we randomly choose a
Ours —> GT —> Shih et al. [37]

Fig. 8: Extrapolation results of our method (left) compared with ground truth (center) and Shih et al. [37] (right).

<table>
<thead>
<tr>
<th>Model</th>
<th>Torch [ms]</th>
<th>TRT-32 [ms]</th>
<th>TRT-16 [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>288p</td>
<td>540p</td>
<td>288p</td>
</tr>
<tr>
<td>F-32</td>
<td>28.7</td>
<td>96.8</td>
<td>27.1</td>
</tr>
<tr>
<td>F-32-NoSM</td>
<td>27.1</td>
<td>94.1</td>
<td>25.9</td>
</tr>
<tr>
<td>F-19</td>
<td>16.3</td>
<td>55.2</td>
<td>14.7</td>
</tr>
<tr>
<td>F-19-NoSM</td>
<td>15.1</td>
<td>51.5</td>
<td>12.1</td>
</tr>
<tr>
<td>F-17-NoSM</td>
<td>13.5</td>
<td>46.3</td>
<td>11.2</td>
</tr>
<tr>
<td>F-13-NoSM</td>
<td>10.7</td>
<td>35.8</td>
<td>9.0</td>
</tr>
</tbody>
</table>

TABLE IV: Inference times of our networks on RTX 3090. We show native PyTorch and TensorRT (TRT) [56] float32/float16 versions. The times do not include PSV generation (1.5 ms @ 540p).

the results of an F-19 variant that is trained with three input views (F-19-3view). We note that inference times and the number of parameters grow with each additional PSV plane. In training and evaluation, we always choose a third frame outside of the two original source frames. The frame is chosen with at least four frames distance to the other frames. Table III shows that FaDIV-Syn benefits from more views, with the mentioned drawbacks.

**Data Efficiency**: FaDIV-Syn generalizes well with significantly smaller training subsets (35%, 5%, 1%). This experiment can be found in our supplementary material.

**C. Inference Time**

Table IV shows the inference time of the different models on one NVIDIA RTX 3090 GPU. Our full model achieves 34fps on 288p input. Removing the soft-masking network results in slightly faster inference speed. Reducing the PSV plane count yields larger gains. We especially note that F-19 is 43% faster than F-32, but attains nearly the same accuracy (see Table I). The fastest model is F-13-NoSM, as it only uses 13 depth planes and no SM network.

In addition to results on vanilla PyTorch, Table IV also shows inference times in TensorRT [56], which already boosts performance for float32 precision without losing accuracy. TensorRT offers possibilities to quantize the weights of neural networks to float16. This quantization of our models retains almost 100% accuracy for SSIM and PSNR and approximately 99% for LPIPS. Further, it leads to a significant acceleration: We can achieve up to 71 fps on 960×540 images and 256 fps for 512×288 resolution. In comparison, 3D-Photo and Stereo-Mag process 2-3 min and 93 ms per 540p image, respectively.

If faster processing times are desired without removing the soft-masking network, one can also estimate the soft-masks in half resolution and upsample them without losing much accuracy (see Table V).

**D. Qualitative Results**

**Continuous Depth**: The fixed plane depths do not constrain FaDIV-Syn. This effect can particularly be seen on straight lines across different depths, which are preserved by our approach (see Fig. 3 and suppl. material). We conclude that FaDIV-Syn does not directly propagate information from the PSV, but properly interpolates between planes.

**Occlusions & Disocclusions**: Figs. 1, 3 and 7 show examples of disocclusions. Our method handles disocclusions for both interpolation and extrapolation. As shown in Figs. 1 and 9, FaDIV-Syn can also handle and represent occlusions.

**Reflections and Transparency**: Often methods that use depth have problems in representing transparencies and reflections (see suppl. material), since there is often more than one depth value at a certain pixel location. FaDIV-Syn is designed

<table>
<thead>
<tr>
<th>Method</th>
<th>SSIM↑</th>
<th>PSNR↑</th>
<th>LPIPS↓</th>
<th>540p [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-32</td>
<td>.9020</td>
<td>29.39</td>
<td>.0556</td>
<td>Torch TRT-32 TRT-16</td>
</tr>
<tr>
<td>F-32 (mixed scale)</td>
<td>.8967</td>
<td>28.99</td>
<td>.0623</td>
<td>96.8 89.2 39.7</td>
</tr>
<tr>
<td>F-19</td>
<td>.8985</td>
<td>29.04</td>
<td>.0583</td>
<td>76.5 71.9 32.8</td>
</tr>
<tr>
<td>F-19 (mixed scale)</td>
<td>.8943</td>
<td>28.74</td>
<td>.0611</td>
<td>55.2 49.3 25.3</td>
</tr>
</tbody>
</table>

TABLE V: Mixed scale inference with 288p mask generation and 576p fusion as described in Sec. IV-C. We show extrapolation performance and timings.
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. 9. Examples for transparencies can be found in Figs. 3 and 8.

Moving Objects: While dynamic scenes are more difficult for view synthesis (and one could argue that the problem is ill-posed), FaDIV-Syn nonetheless shows plausible behavior here. For example, it has learned to interpolate between positions of movable objects (see Fig. 10). This would be hard to achieve if a method estimates depth first or only blends pixels.

E. Limitations

Our approach sometimes struggles with inpainting under large camera movements. One example is the fountain/pool boundary in Fig. 7. We believe that adversarial losses could help to encourage realistic inpainting. Furthermore, we expect that semi-supervised techniques such as proposed by Hani et al. [31] 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 PSV depth distribution is currently fixed and techniques such as depth plane resampling [13] could be advantageous for varying geometries. Another issue we observed is that our network shows very small learning progress after several epochs. While we stop the training after a fixed number of epochs, the validation and testing score still improves for a long time, indicating that further gains are possible through changes in the training regime. Finally, the inference time is limited by the network itself, where compression techniques [57] could be applied to reduce network runtime even further. Note that additional demonstrations of failure cases can be found in the supplementary material.

V. Conclusion

We introduced FaDIV-Syn, a fast depth-independent novel view synthesis method that exceeds state-of-the-art performance in interpolation and extrapolation on the RealEstate10k dataset. The method generalizes well to larger resolutions. Furthermore, its lightweight architecture makes our method real-time-capable with 25-256 fps, depending on output resolution and desired quality. The fast inference times make it applicable for live applications. The proposed gating module encourages self-supervised learning of soft masks, which noticeably improves performance and provides valuable insight into the network operation. Overall, we conclude that the direct usage of the PSV for RGB view synthesis is a promising approach especially for real-time applications and will inspire further research in this direction.

ACKNOWLEDGMENTS

This work was funded by grant BE 2556/16-2 (Research Unit FOR 2535 Anticipating Human Behavior) of the German Research Foundation (DFG) and the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01IS18038C).

REFERENCES


