Boosting View Synthesis with Residual Transfer

Xuejian Rong Jia-Bin Huang Ayush Saraf Changil Kim Johannes Kopf

Reality Labs, Meta
boosting-view-synth.github.io

NeRF [25]
2.5k / 300k iterations
21.38dB / 29.31dB PSNR

Boosted NeRF (ours)
2.5k / 300k iterations
25.29dB / 31.56dB PSNR

NeX [42]
40 / 4,000 epochs
24.87dB / 31.24dB PSNR

Boosted NeX (ours)
40 / 4,000 epochs
29.00dB / 33.14dB PSNR

Figure 1. **Boosting View Synthesis.** We present a simple yet effective approach for injecting residual details (the difference between the ground truth training views and their reconstruction) into the rendering procedure of pre-trained base view synthesis models to boost their rendering quality.

Abstract

Volumetric view synthesis methods with neural representations, such as NeRF and NeX, have recently demonstrated high-quality novel view synthesis. However, optimizing these representations is slow, and even fully trained models cannot reproduce all fine details in the input views. We present a simple but effective technique to boost the rendering quality, which can be easily integrated with most view synthesis methods. The core idea is to transfer color residuals (the difference between the input images and their reconstruction) from training views to novel views. We blend the residuals from multiple views using a heuristic weighting scheme depending on ray visibility and angular differences. We integrate our technique with several state-of-the-art view synthesis methods and evaluate the Real Forward-facing and the Shiny datasets. Our results show that at about 1/10th the number of training iterations, we achieve the same rendering quality as fully converged NeRF and NeX models, and when applied to fully converged models, we significantly improve their rendering quality.

1. Introduction

State-of-the-art neural view synthesis methods such as NeRF [25] and NeX [42], to name just two, have recently shown near-photorealistic novel view synthesis results. These methods model a scene using a 5D radiance field (3 spatial and 2 view direction dimensions) of continuous volumetric density and color. Novel views are synthesized using numerical quadrature for approximating the volume rendering integral. The radiance field is represented using neural methods, particularly a multilayer perceptron (MLP) in the case of NeRF and feature coefficients on a multi-plane image in the case of NeX. Because volume rendering is differentiable, it is straightforward to optimize the parameters of these representations from a sparse set of posed training images by minimizing the difference between these ground truth views and their reconstructions.

However, the training typically requires a long optimization time on the order of several hours to achieve good visual quality. Even a fully trained model struggles with capturing high-frequency details in the training images for complex scenes. Figure 2 shows an example highlighting

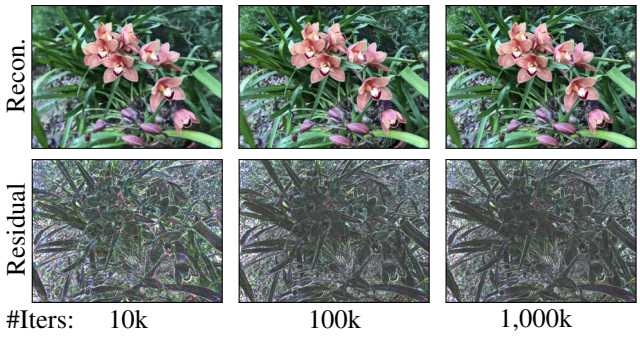


Figure 2. **Reconstructing training images with NeRF.** The training is computationally expensive. However, even a fully trained NeRF model (up to 1 million iterations) cannot recover all fine appearance details present in the training images. The bottom row shows the scaled residual map (difference between the ground truth and the rendered image). Results best viewed on a monitor with zoom-in.

the challenges of reproducing fine details in an input image.

In this paper we present a method to *boost* the rendering quality of many view synthesis methods by injecting these missing details into their volume rendering procedure. Given a pre-trained model, we first render the scene at the input training views and pre-compute *residual* color images, i.e., the difference between the training views and their reconstruction. We then transfer these residual colors from the training views to refine the novel color prediction when synthesizing novel views. More specifically, we retrieve multiple residual colors for each sampled 3D point during volume rendering by projecting it onto the image planes of training views. We then blend them using a heuristic weighting function that takes visibility and angular deviation into account and improves color prediction by adding blended residuals. This results in perfect zero-error reconstructions at training views and significantly improved reconstructions at nearby unobserved views (See Figure 1).

We integrate our approach with several state-of-the-art view synthesis methods, including NeRF and NeX. We evaluate the effectiveness of our method on the Real Forward-facing [25] and Shiny [42] datasets. For most view synthesis methods, integrating our proposed approach requires only a few lines of code changes with and negligible runtime overhead compared to the baseline runtime. Our results show that our approach improves the novel view rendering quality of fully converged baseline methods. Alternatively, it achieves the same quality as fully converged baseline models at significantly reduced training times (e.g., about 1/10th number of iterations in the case of NeRF).

2. Related Work

Image-based rendering. Synthesizing novel views of a scene is a long-standing problem in computer vision [35]. Early approaches using Light Fields [17] or Lumigraphs [10] demonstrate photorealistic rendering by blending and interpolating rays from the collection of captured images without explicit geometric reconstruction. However, they often require capturing an excessive amount of images. Leveraging geometric proxies helps to improve the view synthesis quality without densely sampled camera views [3, 4, 28]. The basic idea is to warp, stitch, and blend the input views to produce the desired target view. Recent work focuses on using learning-based methods for improving blending [11], disparity estimation [8, 14], and view-dependent color synthesis [30, 31].

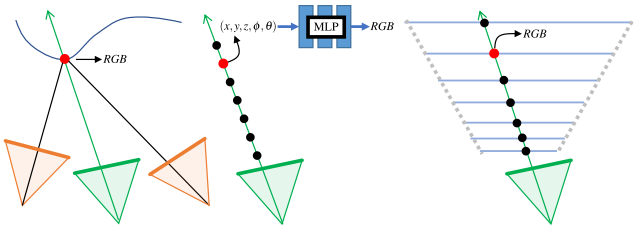
Our method is inspired by classical image-based rendering techniques in the sense that we also transfer content from the training views to the target view. Our work differs in two major aspects. First, instead of transferring color or local image features from the training views, we transfer *residuals* for refining a base view synthesis method. Second, unlike existing methods that perform the warping on surface models, our residual transfer integrates directly into the 3D volume rendering procedure of modern view synthesis methods.

Explicit scene representations. A line of recent research reconstructs 3D (latent) representations from 2D image observations using differentiable renderers. These methods differ in the types of explicit geometric proxies used, including voxels [22, 37], textured mesh [40], multiplane images [7, 24, 42], layered depth images [16, 33, 34], and point clouds [1, 32, 43]. Many of these methods learn latent features in 3D in conjunction with a 2D neural renderer (e.g., a CNN model) that decodes the rendered 2D feature maps to 2D color image. We refer the readers to the recent survey [39] for a thorough discussion.

Implicit scene representations. To overcome limitations on the spatial resolution or additional dimensionalities such as view-dependent appearance, recent work explores encoding scenes with implicit representations [2, 15, 18, 19, 25–27, 38]. The core idea is to map input coordinates (e.g., 3D position and 2D viewing direction) to the desired scene properties (e.g., color, density, signed distance function). In particular, NeRF [25] combines implicit scene representation with volume rendering, demonstrating state-of-the-art view synthesis results.

Our method is *model-agnostic*. We show that the proposed residual transfer can significantly improve the rendering quality of either for implicit representation methods (NeRF [25]) or explicit or hybrid model (NeX [42]).

Extending NeRFs using local image features. To avoid per-scene optimization, several methods extract local features from training images and aggregate them using multi-view consistency to predict color and density [5, 6, 20, 41, 45]. Similar to these methods, our approach also explicitly uses the training images during rendering.



(a) Classic IBR (b) Implicit-based (c) MPI-based

Figure 3. **Example view synthesis methods.** (a) *Classic IBR methods* [3, 11, 28, 31] use a surface-based proxy such as a mesh to aggregate the colors or local features from nearby source views (orange) via reprojection to predict color. (b) *Implicit scene representation* based methods [25] learns a continuous 5D radiance field via an MLP. (c) *MPI-based methods* [7, 24, 42] represent a 3D scene as a stack of fronto-parallel RGBα textures (potentially augmented with view-dependent appearance [42]).

However, our goal is to improve the rendering quality of a pre-trained NeRF (instead of addressing the generalization issue). In contrast to transferring the original colors (or local features) from the training views, we transfer the *residual* to refine the color prediction of an existing view synthesis method.

Residual detail transfer. Neural rendering methods for decoding local image features back to color images often lack fine details. Computing and adding the residual back to the rendered results are simple yet effective techniques for improving the quality [21, 23]. The computed residual can be re-distributed to different layers [23] or different frames in a video [21]. Our work also exploits the idea of residual detail transfer in [21, 23]. Unlike methods that warp/transfer details on *2D image planes* [21, 23], our performs the residual transfer and blending directly *in 3D*.

3. Overview

3.1. View Synthesis

Many classic image-based rendering (IBR) algorithms work by warping and blending input images on a surface-based proxy of the scene geometry (Figure 3a). The blending uses a weighting function that takes factors such as visibility, angular deviation, etc. into account. It can be either hand-crafted [3], or, more recently, learning-based [12, 31].

Since it is difficult to model complex or fuzzy scene details with surface-based models, many recent methods have shifted to using volumetric scene models, typically a 5D radiance field (3 spatial dimensions and 2 for view-dependent appearance). At each position, the field stores RGB color and volume density. Novel views are then synthesized by volume rendering the field. The high dimensional radiance field is typically represented using neural methods, which differ in the machinery for recovering density and color, and where they place samples. For example, NeRF [25] uses an MLP that is adaptively sampled (Figure 3b), while NeX [42] uses an MPI with neural coefficients that is sampled at the fixed multi-planes (Figure 3c).

3.2. Residual Transfer

Because 5D radiance fields are very flexible due to the view-dependent appearance variation, it is trivial to create a radiance field that perfectly reproduces all training images. However, a naïvely fitted radiance field would not generalize well to novel views. So, methods need to balance overfitting the training views and generalizing to novel views. They achieve this through specific priors. For example, the network design of NeRF favors explaining the training views through triangulated geometric variation rather than view-dependent appearance wherever possible. It does this by restricting the density to be a function of only the spatial dimensions and through the compressive force of limiting the network capacity (favoring a "simpler" explanation of the scene where possible). Overall, these methods achieve very high-quality view synthesis. However, they do not perfectly reproduce the training views due to the tradeoff mentioned above. Typically this manifests in fine surface texture details being smoothed out in rendered views (See Figure 1).

The core idea of our method is to evaluate the residual error of the baseline view synthesis method at the training views and then *transfer* these missing details to nearby novel views. We achieve this by leveraging the blending mechanism of classic IBR algorithms to inject the missing residual into the baseline method. We call this *boosting* the algorithm. This simple change to the underlying algorithm typically causes negligible runtime overhead. The resulting boosted view synthesis algorithm can perfectly reconstruct all training views and significantly improve nearby novel views. However, it does not suffer from overfitting artifacts because it only applies the minimum change necessary (i.e., the residual error).

4. Method

As mentioned in the previous section, our goal is to transfer residual details from training views to novel views. We will first explain in Section 4.1 the blending mechanism of classic IBR algorithms, which we use to blend residuals. Then we will recap in Section 4.2 modern volumetric view synthesis methods and make a connection to classic IBR. Finally, we will explain in Section 4.3 how to boost view synthesis algorithms by injecting blended residuals.

4.1. Classic Surface-based View Synthesis

Classic IBR methods transfer *colors* from the input views to novel views by blending them on a surface-based proxy of the scene. We will later use a similar mechanism

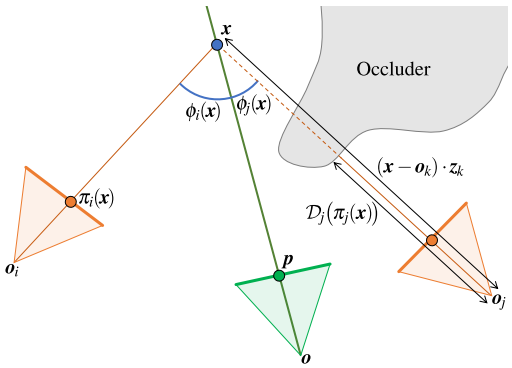


Figure 4. **Blending weights for residual transfer.** Here we show an example with just two training views for clarity. For rendering a ray through pixel $\boldsymbol{p}$ for a virtual camera located at $\boldsymbol{o}$, we sample 3D points along the ray. For each 3D point (e.g., $\boldsymbol{x}$), we project it onto the image planes of the training views to obtain the corresponding 2D image coordinates $\pi_i(\boldsymbol{x})$ and $\pi_j(\boldsymbol{x})$.We retrieve the residual colors from each view, $\mathcal{R}_i(\pi_i(\boldsymbol{x}))$ and $\mathcal{R}_j(\pi_j(\boldsymbol{x}))$, and blend these residual colors depending on their angular differences ϕ_i and ϕ_j.

for blending *residuals*, but it is helpful to discuss the classic case with color transfer first because it allows us to establish some useful notation and math.

The input is a set of images $\{\mathcal{I}_k\}_{k=1}^K$ with camera poses $P_k = (\boldsymbol{K}_k, \boldsymbol{R}_k, \boldsymbol{t}_k)$, i.e., intrinsics, rotation, and translation, respectively. Given a new pose $P = (\boldsymbol{K}, \boldsymbol{R}, \boldsymbol{t})$, the goal is to synthesize an image $\mathcal{O}$ that shows a view of the scene from that viewpoint.

The output color at a pixel $\boldsymbol{p}$ is computed as a weighted combination of pixels from the input views:

$$\mathcal{O}(\boldsymbol{p}) = \sum_{k=1}^{K} w_k(\boldsymbol{x})\, \mathcal{I}_k\big(\pi_k(\boldsymbol{x})\big). \tag{1}$$

Here, $\boldsymbol{x}$ is the point at which the ray through $\boldsymbol{p}$ intersects with the surface proxy, and π_k computes the projection into the k-th input view:

$$\pi_k(\boldsymbol{x}) = \mu\big(\boldsymbol{K}_k(\boldsymbol{R}_k\boldsymbol{x} + \boldsymbol{t}_k)\big), \tag{2}$$

with $\mu\left([x,y,z]^\top\right) = \left[\frac{x}{z}, \frac{y}{z}\right]$.

The blending weights we use contain a visibility term v that disables input pixels that cannot see $\boldsymbol{x}$ and an angular term ϕ that favors input views that observe $\boldsymbol{x}$ from a similar direction as the novel view:

$$w_k(\boldsymbol{x}) = \frac{1}{W} \cdot \frac{v_k(\boldsymbol{x})}{\phi_k(\boldsymbol{x}) + \varepsilon}. \tag{3}$$

with W is for normalization so that $\sum w_k(\boldsymbol{x}) = 1$, and we use Softmax to function as the $1/W$ term in experiments. A very small positive ε avoids division by zero.

In practice, we observe that blending a small set of the most relevant training views leads to better results than blending all training views. Accordingly, we set all weights among $\{w_k\}$ that are not in the top $T = 5$ to zero, and then use W to renormalize the top T weights to sum to one.

The visibility term softly down weighs pixels where $\boldsymbol{x}$ is occluded:

$$v_k(\boldsymbol{x}) = 1 - S\left(\frac{(\boldsymbol{x} - \boldsymbol{o}_k)\cdot \boldsymbol{z}_k}{\mathcal{D}_k\big(\pi_k(\boldsymbol{x})\big)} - 1\right). \tag{4}$$

Here, $\boldsymbol{z}_k = \boldsymbol{R}_k^\top \cdot [0,0,1]^\top$ is the principal direction of the k-th view, and $\boldsymbol{o}_k = -\boldsymbol{R}_k^\top \boldsymbol{t}_k$ is the camera center. The term in the numerator is the z-depth and the denominator samples a precomputed depth map $\mathcal{D}_k$. $S(t) = \left(1 + e^{-k(t-t_0)}\right)^{-1}$ is the logistic function (a sigmoid), $k = 50$ makes the sigmoid steep and edge-like, and $t_0 = 0.1$ shifts it slightly, so that we include samples that are *slightly* occluded due to inaccurately estimated depth ($v_k(\boldsymbol{x})$ fades from 1 to 0 for depth ratios between approximately 1.0 and 1.2).

The angular term $\phi_k(\boldsymbol{x})$ measures the angle between the novel view ray and the k-th input view ray:

$$\phi_k(\boldsymbol{x}) = \measuredangle\big(\overrightarrow{\boldsymbol{x}\boldsymbol{o}}, \overrightarrow{\boldsymbol{x}\boldsymbol{o}_k}\big). \tag{5}$$

Note that when the output pose approaches one of the input views, the respective angular term vanishes, and the weights become one-hot, so the input view is perfectly reconstructed. Figure 4 illustrates the notation in this section on a schematic example.

4.2. Volumetric View Synthesis

It is difficult to accurately represent complex scene details with surface-based proxies because surfaces are hard to estimate in smooth regions, complex surface topology is challenging to optimize, and surface-based appearance does not well support semi-transparency and reflections. For this reason, more recent methods, such as NeRF [25] or NeX [42], replace the surface-based models with volumetric approaches where volume density varies continuously.

In this setting, the output image is obtained by volume rendering. The output color at a pixel $\boldsymbol{p}$ is obtained by integrating color $\boldsymbol{c}$ and density σ along the corresponding ray $\boldsymbol{r}(\boldsymbol{p}, t) = \boldsymbol{o} + t\boldsymbol{d}(\boldsymbol{p})$:

$$\mathcal{O}(\boldsymbol{p}) = \int_{t_n}^{t_f} T(t)\, \underbrace{\sigma\big(\boldsymbol{r}(\boldsymbol{p},t)\big)}_{\text{Density}}\, \underbrace{\boldsymbol{c}\big(\boldsymbol{r}(\boldsymbol{p},t), \boldsymbol{d}(\boldsymbol{p})\big)}_{\text{Emitted color}}\, \mathrm{d}t. \tag{6}$$

The second and third term are the *local* density and emitted color, respectively, at the depth t along the ray, and $T(t)$ is the *accumulated* transmittance to this point:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma\big(\boldsymbol{r}(\boldsymbol{p},s)\big)\, \mathrm{d}s\right). \tag{7}$$

In practice, the volume rendering integral in Equation 6 is approximated with a sum. To this end, we sample density and color $\{\sigma_i, \boldsymbol{c}_i\}_{i=1}^N$ at depths $\{t_i\}_{i=1}^N$ along the ray. The discrete version of volume rendering then becomes:

$$\mathcal{O}(\boldsymbol{p}) = \sum_{i=1}^{N} T_i \, \alpha_i \, \boldsymbol{c}_i \,, \tag{8}$$

with:

$$T_i = \exp\Big(-\sum_{j=1}^{i-1} \sigma_j \delta_j\Big), \text{ and } \alpha_i = 1 - \exp(-\sigma_i \delta_i) \,. \tag{9}$$

Here, $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. Evaluating Equation 8 boils down to simple alpha compositing.

Different volumetric view synthesis methods vary in how they represent density and color and how the volume is sampled. For example, NeRF uses an MLP that is adaptively sampled. NeX uses a multi-plane image (i.e., fixed sample positions) and stores neural coefficients transformed into color and density using a small network in rendering.

4.3. Boosting View Synthesis via Residual Transfer

We can combine classic blending (Section 4.1) and volumetric view synthesis (Section 4.2) by injecting the color blending from Equation 1 into the volume rendering in Equation 8:

$$\mathcal{O}(\boldsymbol{p}) = \sum_{i=1}^{N} T_i \, \alpha_i \underbrace{\sum_{k=1}^{K} w_k(\boldsymbol{x}_i) \, \mathcal{I}_k\big(\pi_k(\boldsymbol{x}_i)\big)}_{\text{Blending (Equation 1)}}, \tag{10}$$

with $\boldsymbol{x}_i = \boldsymbol{r}(\boldsymbol{p}, t_i)$. This equation accumulates blended colors from training views along the ray using the reconstructed volume density of a baseline view synthesis method, for example, NeRF or NeX. Like classic IBR, it reconstructs all training views perfectly with zero error, but it provides superior quality for novel views compared to classic IBR because the continuous volume density is a better scene proxy than surface-based models.

On the other hand, the underlying baseline view synthesis method does *not* reconstruct the training views perfectly, for the reasons described in Section 3.2, which often manifests in smoothed-out surface details. But it typically has superior generalization ability than Equation 10 due to the more sophisticated optimization that went into precomputing the color volume.

This is where the core idea of this paper comes into play. It is to combine the strengths of both approaches: the high generalization ability of the baseline view synthesis method *and* the ability to reconstruct most visible details in the training views. We achieve this by splicing Equations 8 and 10:

$$\mathcal{O}(\boldsymbol{p}) = \sum_{i=1}^{N} T_i \, \alpha_i \Big(\boldsymbol{c}_i + \underbrace{\sum_{k=1}^{K} w_k(\boldsymbol{x}_i) \, \mathcal{R}_k\big(\pi_k(\boldsymbol{x}_i)\big)}_{\text{Blended residual}}\Big), \tag{11}$$

where instead of blending colors we blend the residual errors,

$$\mathcal{R}_k = \mathcal{I}_k - \mathcal{O}_{|P=P_k} \,. \tag{12}$$

The blended residual adds in just the missing detail that the baseline view synthesis method was not able to reconstruct by itself. This results in perfect reconstruction of training views and significantly improved surrounding novel views. We call this *boosting* the underlying view synthesis method.

Ghosting is limited because boosting only adds in residuals, i.e., the minimum change needed for a better reconstruction. Since evaluating Equation 11 does not require any additional network evaluation compared to the baseline (Equation 8), the runtime overhead of boosting is often negligible compared to the baseline.

The full method has the following parameters: $T = 5$ (number of blended residuals), $k = 50$, $t_0 = 0.1$ (sigmoid shape parameters). Throughout all experiments, we use these fixed values.

5. Experimental Results

This section validates the proposed residual transfer method for boosting the rendering quality of existing view synthesis methods. We first describe our experimental setup (Section 5.1), and then report the quantitative evaluation of applying our method to two state-of-the-art view synthesis methods (Section 5.2). We show sample visual results to highlight the perceptual improvement (Section 5.3) and validate our core design choices (Section 5.4). We will release the source code for the full reproducibility of our method.

We include sample visual results and report the main quantitative evaluation on the datasets in the following. Many visual improvements are most visible when viewed at high resolution on a monitor. We encourage the readers to see our supplementary material, where we include extensive view synthesis results and visual comparisons.

5.1. Experimental Setup

Dataset. We evaluate our method on the Real Forward-facing [25] and Shiny [42] datasets. Each dataset consists of 8 scenes captured using a smartphone. Many of these challenging scenes contain complex scene geometry (e.g., thin structures), semi-transparent surfaces (e.g., windows), and strong view-dependent effects (e.g., shiny reflective objects). We use the default training and testing split from the respective dataset to conduct our evaluation.

Table 1. **Quantitative comparisons.** We report the results using the NeRF model trained with 300k steps and the NeX models trained for 16 hours on 4 NVIDIA V100 GPUs. We aggregate the metrics over all the scenes for the Real Forward-facing dataset [25] and the Shiny dataset [42]. We use VGGNet [36] for computing the LPIPS [46] scores.

Methods	Real Forward-facing [24]			Shiny dataset [42]		
	PSNR↑	SSIM↑	LPIPS ↓	PSNR↑	SSIM↑	LPIPS ↓
NeRF [25]	26.53	0.8038	0.1985	22.43	0.6360	0.3018
Boosted NeRF (ours)	27.08	0.8371	0.1383	23.22	0.7024	0.1975
NeX [42]	27.74	0.8702	0.1302	26.74	0.8321	0.1293
Boosted NeX (ours)	28.19	0.8723	0.1104	28.09	0.8642	0.0963

Table 2. **Per-scene PSNR comparisons on the Real Forward-Facing dataset [25]**. We use NeRF model trained with 300k steps as our base model.

Scene	NeRF [25]	Boosted NeRF (ours)
Fern	25.07	25.18 (+0.11)
Flower	26.64	28.69 (+2.05)
Fortress	31.39	32.10 (+0.71)
Horns	27.68	28.52 (+0.84)
Leaves	20.98	21.14 (+0.16)
Orchids	20.34	20.41 (+0.07)
Room	32.20	32.94 (+0.74)
Trex	26.91	27.63 (+0.72)

Table 3. **Per-scene PSNR comparisons on the Shiny dataset [42]**. We use NeX model trained with 4,000 epochs (2,000 for `cd` and `lab`) as our base model.

Scene	NeX [25]	Boosted NeX (ours)
Cd	30.38	32.49 (+2.11)
Crest	22.43	22.90 (+0.47)
Food	24.75	25.32 (+0.57)
Giants	27.48	28.41 (+0.93)
Lab	30.12	31.74 (+1.62)
Pasta	22.94	24.23 (+1.29)
Seasoning	29.99	30.40 (+0.41)
Tools	29.16	29.25 (+0.09)

Table 4. **Ablation study on the *Horns* scene.** Simply transferring the blended *colors* from the training views performs poorly. In contrast, transferring the blended *residual* substantially improve the rendering quality of the baseline NeRF model.

Methods	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline NeRF [25]	27.68	0.8226	0.2142
Color transfer with Eqn. (10)	23.84	0.8477	0.1753
Residual transfer with Eqn. (11)	28.52	0.8801	0.1230

Compared methods. As our method is *model-agnostic*, we choose two representative volumetric view synthesis algorithms, NeRF [25] and NeX [42], as our base models. For NeRF, we use the PyTorch implementation[1]. For NeX, we use the official implementation provided by the authors[2]. Note that when using the NeX model, we follow the instruction and use the *undistorted* version of the Real Forward-facing [25] dataset as provided by the authors of NeX [42]. We follow the default hyper-parameters in both methods for all of our experiments. Following the respective papers, the NeRF models were trained for 300k iterations and NeX models were trained for 4,000 epochs (2,000 for `cd` and `lab` due to the large size) on 4 NVIDIA V100 GPUs.

As our goal is *not* to develop a new view synthesis method, in the following we do not include additional comparisons with other view synthesis methods such as LLFF [24], SRN [38], or NSVF [18]. Instead, we focus on the evaluation of using our method to boost the two selected base models.

5.2. Quantitative Comparison

Improvement over fully trained models. Table 1 summarizes the quantitative comparisons on the *test* splits of the two datasets. Adding the proposed residual transfer for both base methods leads to clear improvement even for fully trained models. Also, our method (trivially) achieves perfect zero-error reconstruction at training views. In Table 2 and Table 3, we provide per-scene quantitative comparison with the NeRF and NeX models.

Improvement at different training stages. Optimizing these base models often involves long training time. Figure 6 shows the rendering quality improvement over the base NeRF models with a different number of training iterations. Our results show that we can achieve the same quality as fully converged NeRF models with about 1/10th of the total training iterations.

5.3. Visual Comparison

Figure 5 shows several view synthesis results on the Real Forward-facing and the Shiny datasets. Our method provides substantial visual improvement by injecting the missing high-frequency details from the training views into novel views. Please refer to the supplementary material for additional visual comparisons.

[1] https://github.com/yenchenlin/nerf-pytorch
[2] https://github.com/nex-mpi/nex-code/

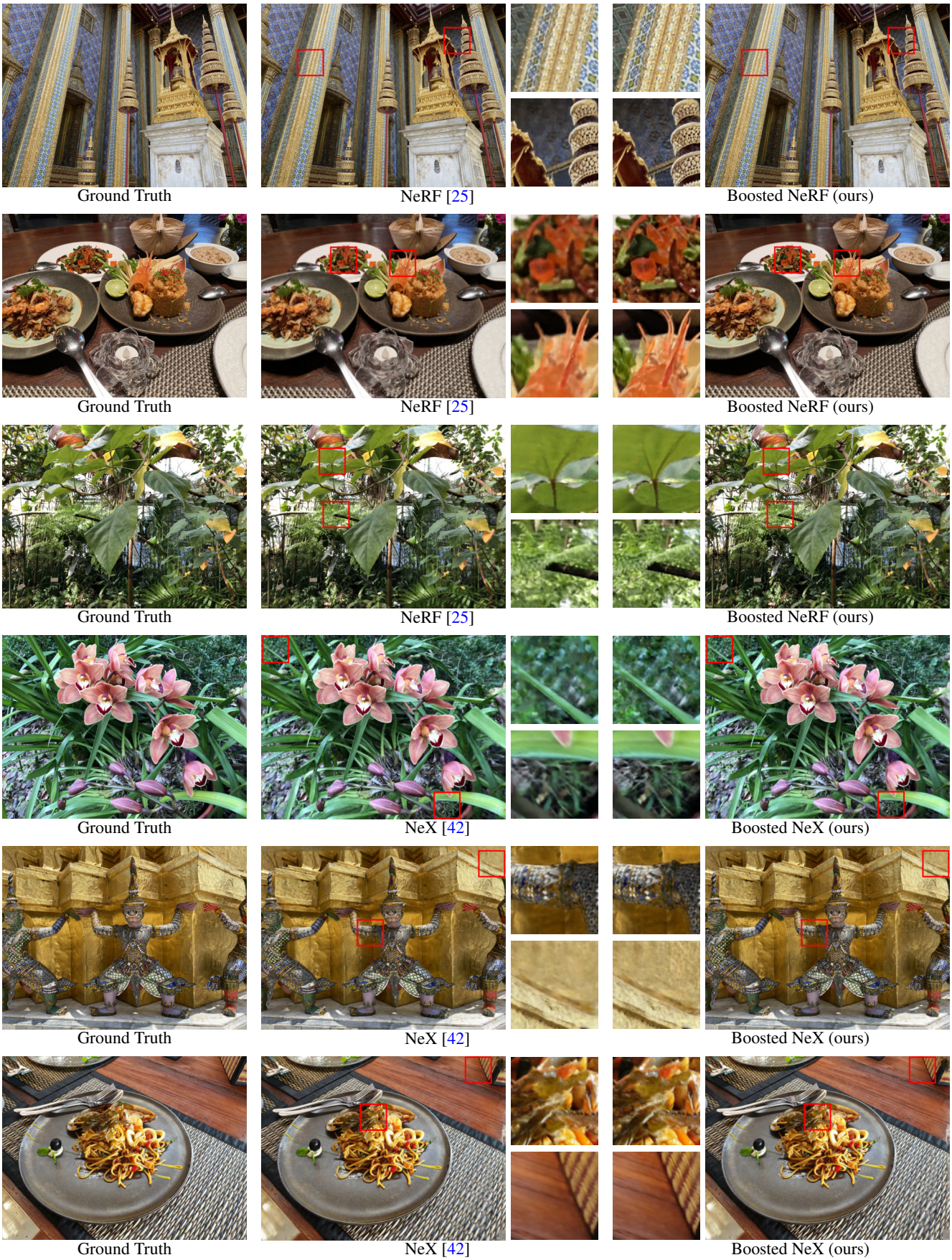


Figure 5. **Visual comparisons** of the boosted results with the baseline NeRF [25] (*top 3 rows*) and NeX [42] (*bottom 3 rows*).

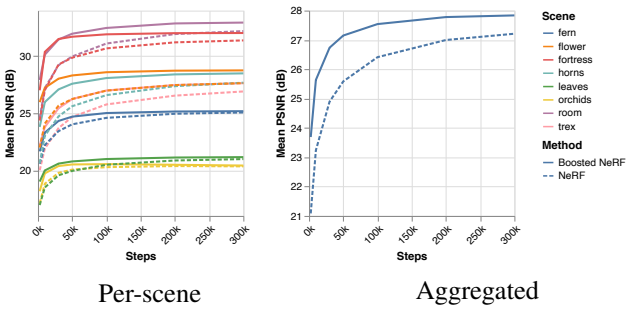


Figure 6. **Quantitative evaluation over the number of iterations.** We present the quantitative results on the *test* split of the Real Forward-facing [25]. We report the averaged PSNR values under different checkpoints during the course the training. Our residual transfer achieves significant improvement over NeRF at early stage of training and moderate improvement for fully trained model. Rendering the boosted NeRF at *training* views (trivially) produces perfect reconstruction.

5.4. Ablation Study

We show in Table 4 the ablation of using either *color* transfer (similar to classical IBR approaches) or the proposed *residual transfer*. Our results show that while using the same density provided by a trained NeRF model, volume rendering using the transferred colors leads to substantially inferior results than the baseline NeRF model. In contrast, our residual transfer approach results in a clear PSNR improvement for the *Horns* scene.

5.5. Computational Overhead

Using a single NVIDIA V100 GPU, rendering a frame of spatial resolution 1008×756 takes around 16 seconds with NeRF [25] (measured by averaging over 15 runs), as demonstrated in Table 5. In comparison, incorporating our proposed residual transfer method incurs only a few percentages of the original computation time for rendering, i.e. approximately two orders of magnitude smaller than the baselines, when we properly factor the residual blending out of the time-consuming volumetric rendering pipeline as a post-processing scheme.

5.6. Memory allocation Overhead

Besides the trained weights of the baseline NeRF, our method would require storing additional residuals of reference GT views for blending, and we conducted the ablation study for the memory allocation in Table 5 on the same GPU setup. The additional memory footprint for storing the raw color information of reference residuals is proportional to the count of utilized GT views, which is substantial when uncompressed. We believe using a sparse set (e.g., via ranking) of the raw residual maps or corresponding features, with a heuristic or learnable fusion mechanism can potentially retain similar improvement while substantially reducing the memory requirement.

Table 5. **Ablation study on computational and memory footprint overhead on the *Horns* scene.** We logged the computational time and memory allocation (average of 15 runs) on rendering one 1008x756 novel view, to compute the corresponding overhead in terms of utilizing increasing numbers of reference GT views.

Methods	Comp. time (sec)	Memory (MB)	PSNR ↑
Baseline NeRF [25]	16.34	4.79	27.68
Ours (1 ref. view)	16.56 (+1.35%)	7.08 (+47.81%)	27.94 (+0.26)
Ours (3 ref. view)	16.63 (+1.77%)	11.66 (+143.42%)	28.36 (+0.68)
Ours (5 ref. view)	16.68 (+2.08%)	16.24 (+239.04%)	28.52 (+0.84)
Ours (10 ref. view)	16.74 (+2.45%)	27.69 (+478.08%)	28.49 (+0.81)

6. Limitations

Memory. Compared to neural radiance field-based methods such as NeRF [25] that encode the scene compactly as the weights of an MLP, our method incurs additional memory requirements because we need to store the residual images of the training views.

Computational overhead. Our method involves simple, easily parallelizable operations such as 3D point projection onto training views, interpolation to obtain residual values, and addition to refining the estimated colors from view synthesis methods. These additional computations are negligible for NeRF rendering. However, there have been many recent work to speedup the rendering via baking [13], caching [9], or space partitioning [29, 44]. The computational overhead for our method (without careful engineering) may no longer be negligible for these fast methods.

Ghosting artifacts. Our method relies on the accurate volume density or proxy geometry (estimated by view synthesis methods) to properly transfer the residuals from the training views. Our method may introduce ghosting artifacts when the density/geometry is erroneous (e.g., insufficient training or limited training views).

Potential negative societal impacts. Our method still relies on base view synthesis models, often computationally expensive and consume significant electricity.

7. Conclusions

We have presented a simple method for boosting the rendering quality of view synthesis methods. Building upon the basic idea in classic image-based rendering algorithms, we adaptively transfer color residuals from the training views to the target novel views. We show that our can be easily integrated with state-of-the-art view synthesis algorithms, including NeRF and NeX, and demonstrate clear visual and quantitative improvement on the Real Forward-facing and the Shiny datasets.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 2

[2] Benjamin Attal, Jia-Bin Huang, Michael Zollhoefer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding networks. In *CVPR*, 2022. 2

[3] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001. 2, 3

[4] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013. 2

[5] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 2

[6] 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

[7] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, 2019. 2, 3

[8] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016. 2

[9] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 8

[10] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 2

[11] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2, 3

[12] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 2018. 3

[13] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021. 8

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

[15] Petr Kellnhofer, Lars Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *CVPR*, 2021. 2

[16] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3d photography. *ACM Transactions on Graphics (TOG)*, 39(4):76–1, 2020. 2

[17] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 2

[18] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2, 6

[19] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020. 2

[20] 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. *arXiv preprint arXiv:2107.13421*, 2021. 2

[21] Yu-Lun Liu, Wei-Sheng Lai, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Hybrid neural fusion for full-frame video stabilization. In *ICCV*, 2021. 3

[22] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. In *SIGGRAPH*, 2019. 2

[23] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video. *ACM Transactions on Graphics (TOG)*, 2020. 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. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. 2, 3, 6

[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, 3, 4, 5, 6, 7, 8

[26] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 2

[27] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021. 2

[28] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 2, 3

[29] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 8

[30] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 2

[31] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 2, 3

[32] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635*, 2021. 2

[33] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998. 2

[34] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2

[35] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, volume 4067, pages 2–13, 2000. 2

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[37] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *CVPR*, 2019. 2

[38] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 2, 6

[39] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727, 2020. 2

[40] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM TOG*, 2019. 2

[41] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[42] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 1, 2, 3, 4, 5, 6, 7

[43] Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. Multi-view neural human rendering. In *CVPR*, 2020. 2

[44] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *CVPR*, 2021. 8

[45] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. Pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[46] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6