

# PaletteNeRF: Palette-based Appearance Editing of Neural Radiance Fields

Zhengfei Kuang<sup>1\*</sup>, Fujun Luan<sup>2</sup>, Sai Bi<sup>2</sup>, Zhixin Shu<sup>2</sup>, Gordon Wetzstein<sup>1</sup>, Kalyan Sunkavalli<sup>2</sup>  
<sup>1</sup>Stanford University      <sup>2</sup>Adobe Researchh

{zhengfei, gordonwz}@stanford.edu      {fluan, sbi, zshu, sunkaval}@adobe.com

<https://palettenerf.github.io>

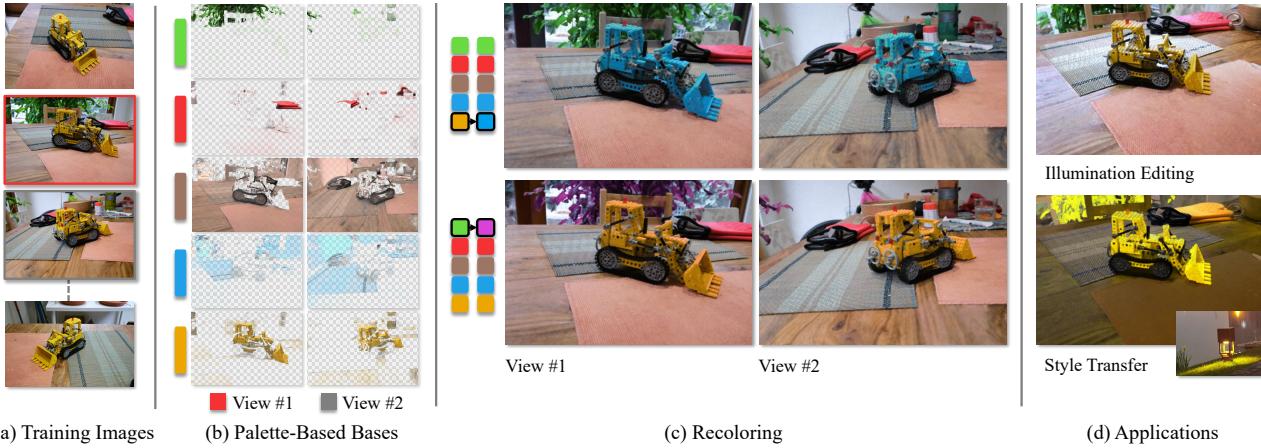


Figure 1. We propose **PaletteNeRF**, a novel method for efficient appearance editing of neural radiance fields (NeRF). Taking (a) multi-view photos as training input, our approach reconstructs a NeRF and decomposes its appearance into a set of (b) 3D palette-based color bases. This enables (c) intuitive and photorealistic recoloring of the scene with 3D consistency across arbitrary views. Further, we show that (d) our method supports various palette-based editing applications such as illumination modification and 3D photorealistic style transfer.

## Abstract

Recent advances in neural radiance fields have enabled the high-fidelity 3D reconstruction of complex scenes for novel view synthesis. However, it remains underexplored how the appearance of such representations can be efficiently edited while maintaining photorealism. In this work, we present **PaletteNeRF**, a novel method for photorealistic appearance editing of neural radiance fields (NeRF) based on 3D color decomposition. Our method decomposes the appearance of each 3D point into a linear combination of palette-based bases (i.e., 3D segmentations defined by a group of NeRF-type functions) that are shared across the scene. While our palette-based bases are view-independent, we also predict a view-dependent function to capture the color residual (e.g., specular shading). During training, we jointly optimize the basis functions and the color palettes,

and we also introduce novel regularizers to encourage the spatial coherence of the decomposition. Our method allows users to efficiently edit the appearance of the 3D scene by modifying the color palettes. We also extend our framework with compressed semantic features for semantic-aware appearance editing. We demonstrate that our technique is superior to baseline methods both quantitatively and qualitatively for appearance editing of complex real-world scenes.

## 1. Introduction

Neural Radiance Fields (NeRF) [23] and its variants [8, 25, 27, 39] have received increasing attention in recent years for their ability to robustly reconstruct real-world 3D scenes from 2D images and enable high-quality, photorealistic novel view synthesis. However, such volumetric representations are challenging to edit due to the fact that scene appearance is implicitly encoded in neural features and net-

\*Parts of this work were done when Zhengfei Kuang was an intern at Adobe Research.

work weights that do not support local manipulation or intuitive modification.

Multiple approaches have been proposed to support editing of NeRF. One category of methods [4, 18, 41, 45] recover the material properties of the scene so that they can re-render them under novel lighting conditions or adjust the material properties such as surface roughness. Such methods rely on accurate estimation of the scene reflectance, which is typically challenging for real-world complex scenes captured under unconstrained environment. Another category of methods [21, 35] learns a latent code on which NeRF can be conditioned to produce the desired appearance. However, these methods often suffer from limited capacity and flexibility and do not support fine-grained editing. In addition, some other methods [40] learn to transfer the appearance of NeRF to match a given style image, but sometimes fail to maintain the same level of photorealism in the original scene.

In this paper, we propose *PaletteNeRF*, a novel method to support flexible and intuitive editing of NeRF. Our method is inspired by previous image-editing methods based on color palettes [7, 31], where a small set of colors are used to represent the full range of colors in the image. We model the radiance of each point using a combination of specular and diffuse components, and we further decompose the diffuse component into a linear combination of view-independent color bases that are shared across the scene. During training, we jointly optimize the per-point specular component, the global color bases and the per-point linear weights to minimize the difference between the rendered images and the ground truth images. We also introduce novel regularizers on the weights to encourage the sparseness and spatially coherence of the decomposition and achieve more meaningful grouping. With the proposed framework, we can intuitively edit the appearance of NeRF by freely modifying the learned color bases (Fig. 1). We further show that our framework can be combined with semantic features to support semantic-aware editing. Unlike previous palette-based image [1, 31] or video [10] editing methods, our method produces more globally coherent and 3D consistent recoloring results of the scene across arbitrary views. We demonstrate that our method can enable more fine-grained local color editing while faithfully maintaining the photorealism of the 3D scene, and achieves better performance than baseline methods both quantitatively and qualitatively. In summary, our contributions include:

- We propose a novel framework to facilitate the editing of NeRF by decomposing the radiance field into a weighted combination of learned color bases.
- We introduced a robust optimization scheme with novel regularizers to achieve intuitive decompositions.

- Our approach enables practical palette-based appearance editing, making it possible for novice users to interactively edit NeRF in an intuitive and controllable manner on commodity hardware.

## 2. Related Work

**Neural radiance fields.** Neural radiance fields in the form of MLPs have been extensively used for neural rendering tasks such as novel view synthesis. Typically, these methods [20, 23, 28, 37] encode the geometry and appearance of the scene into network weights of the MLPs. Many recent works [8, 25, 36, 39, 44] propose to speed up the training and improve the performance of the models by applying a combination of light-weight MLPs and neural feature maps or volumes. However, different from traditional graphics primitives such as triangle meshes, both the neural features and the network weights represent scene appearance in an implicit manner and do not support intuitive editing or controls such as recoloring, thereby greatly limiting their applications in existing graphics pipelines.

**Appearance editing with NeRF.** Many methods have been proposed to support appearance editing of NeRF. Some methods [4, 5, 29, 45] recover the physical properties of the scene such as albedo, specular roughness, and then they can support rendering the scene under novel lighting conditions or changing its material properties. Some other works [15, 21, 35] learn a latent code jointly with the NeRF reconstruction so as to control its appearance such as changing the illuminations or colors by taking new latent codes as input, which can be mapped from user input or interpolated from existing latent codes. Moreover, there are also approaches [9, 13, 40] that try to optimize the NeRF to match its appearance against the provided style images. All these methods do not support fine-grained intuitive color editing of NeRF as ours. Concurrently, Ye et al. [38] introduces a NeRF-based intrinsic decomposition model which enables 3D intuitive recoloring, but it does not support palette-based editing.

**Palette-based editing.** Color palette-based methods [7, 12, 31, 32, 42, 46] have been previously used in 2D image editing tasks such as recoloring. However, it is non-trivial to adapt such methods for NeRF editing and naively performing the editing on each rendered frame cannot guarantee view-consistent results. Instead, we integrate the learning of the color bases and the spatially varying weights into the NeRF optimization process, thereby enabling view-consistent and 3D-aware editing. A recent work [33] proposes a palette-based recoloring method on NeRF with posterization. However, it is limited to non-photorealistic editing only.

### 3. Method

Fig. 2 illustrates the overview of our multi-stage pipeline. Given a set of images with known poses from a scene, we first optimize a NeRF-based model to reconstruct the geometry of the scene. Then, we extract  $N_p$  color palettes with the input images and the learned scene geometry. Finally, we train a segmentation model to decompose the scene appearance into multiple bases based on the extracted palettes. Our decomposition result is able to drive various downstream applications, such as recoloring, photorealistic style transfer and illuminance modification.

#### 3.1. Volumetric Rendering

We build our model on the widely used framework Neural Radiance Fields (NeRF). Typically, a NeRF-based model optimizes two neural functions: a geometry function  $\sigma(\mathbf{x})$  and a color function  $\mathbf{c}(\mathbf{x}, \mathbf{d})$ . The geometry function takes a 3D position as input and outputs the density at this point. The color function takes a 3D position and a viewing direction as input and output a corresponding RGB color. To render an image from a given camera pose, NeRF samples batches of rays from the camera to render the pixels. For each sampled ray  $\mathbf{r} = (\mathbf{o}, \mathbf{d})$ , a group of 3D points are sampled along the ray path  $\mathbf{x}_{1..M}$  with depth  $t_{1..M}$ , where  $\mathbf{x}_i = \mathbf{o} + t_i \mathbf{d}$ . The color prediction of the ray  $\mathbf{r}$  is calculated as:

$$\mathbf{c}(\mathbf{r}) = \sum_{i=1}^M \alpha_i (1 - w_i) \mathbf{c}(\mathbf{x}_i, \mathbf{d}), \quad (1)$$

where  $w_i = \exp(-(t_i - t_{i-1})\sigma(\mathbf{x}_i))$  is the transmittance of the ray between the  $i$ 'th sample point and the  $(i+1)$ 'th sample point, and  $\alpha_i = \prod_{j=1}^{i-1} w_j$  is the ray attenuation from the ray's origin to the  $i$ 'th sample point.

#### 3.2. Palette Extraction

Although it is possible to directly train our model with randomly initialized color palettes, doing this will bring excessive ambiguity to the problem and may produce irregular results as shown in Fig. 3(b). Thankfully, the problem of extracting color palettes from images has been extensively researched over the past years, thus we use the extraction method from a state-of-the-art image recoloring work [31] as our initialization. In general, this method extracts color palettes from the 3D convex hull of the clustered image colors in the RGB space. In our scenario, we simply select all training image pixels with valid depth from NeRF's depth maps and concatenate their color as input.

We notice that with input from multiple images, the extraction method may produce palettes that are chromatically similar (e.g., palettes including a light yellow and a dark yellow), due to the varying shading of the captured scene. This may result in unrealistic appearance editing. Inspired

by image illumination decomposition works [6, 22], we normalize the input colors of the training images by their intensity, to narrow down the search space of color palettes. Conventionally, the intensity of a color is represented by the L1 norm of its RGB value. However, normalizing with L1 norm projects the colors to a plane, which is a highly ill-posed edge case for 3D convex hull calculation. These problems can be addressed by replacing the norm with a higher order one. Empirically, we find that palettes extracted from L2 normalized images work well in our next decomposition stage, and use them in all of our experiments.

In addition to the extracted palettes  $\bar{\mathcal{P}}$ , we also keep the blending weights  $\bar{\omega}$  of the input pixel colors, calculated from the method of the same work. These weights act as an additional supervision in the next stage.

#### 3.3. Color Decomposition

To introduce our color decomposition model, we first explain our target outputs, then describe the structure of our decomposition network.

As shown in Fig. 2, given the color palettes of size  $N_p$ , our model aims to reconstruct  $N_p$  view-independent palette-based bases and an additional view-dependent color function representing all view-dependent shading such as specular reflections. The palette-based bases correspond to the extracted color palettes, and are defined by two functions of  $\mathbf{x}$ : A color offset function  $\delta : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ , and a weight function  $\omega : \mathbb{R}^3 \rightarrow [0, 1]$ . Observing that realistic image captures usually consists of a huge variety of colors, we allow the basis color of each point shift from the palette color with an offset (inspired by image-based soft color segmentation methods [1, 2]). This design increases the capacity of the bases and benefits the segmentation quality on complex scenes. We also introduce an intensity function  $I : \mathbb{R}^3 \rightarrow [0, 1]$  that is shared among all palette-based bases, due to the normalization of the extracted palettes. While the aforementioned functions only take position as input, our model also contains a view-dependent color function  $s : \mathbb{R}^5 \rightarrow [0, 1]^3$  that takes viewing direction as input, too. We compose all of these bases into a color output  $\mathbf{c} \in \mathbb{R}^3$  by the following equation:

$$\mathbf{c}(\mathbf{x}, \mathbf{d}) = s(\mathbf{x}, \mathbf{d}) + I(\mathbf{x}) \sum_{i=1}^{N_p} \omega_i(\mathbf{x}) (\mathcal{P}_i + \delta_i(\mathbf{x})), \quad (2)$$

where  $\omega_i(\mathbf{x})$  are normalized by their sum. We clamp the summed-up color  $\mathbf{c}$  to  $[0, 1]$  as the final output. We also optimize the palette color  $\mathcal{P}_i$  during the training. As shown in Fig. 4, Our network consists of three MLPs: the Diffuse MLP predicts the diffuse color  $\mathbf{c}_d(\mathbf{x})$ , i.e. the sum of all palette-based bases; the View-Dependent MLP generates view-dependent color  $s(\mathbf{x}, \mathbf{d})$ ; Finally, the Palette MLP predicts the function values of the palette-based bases:

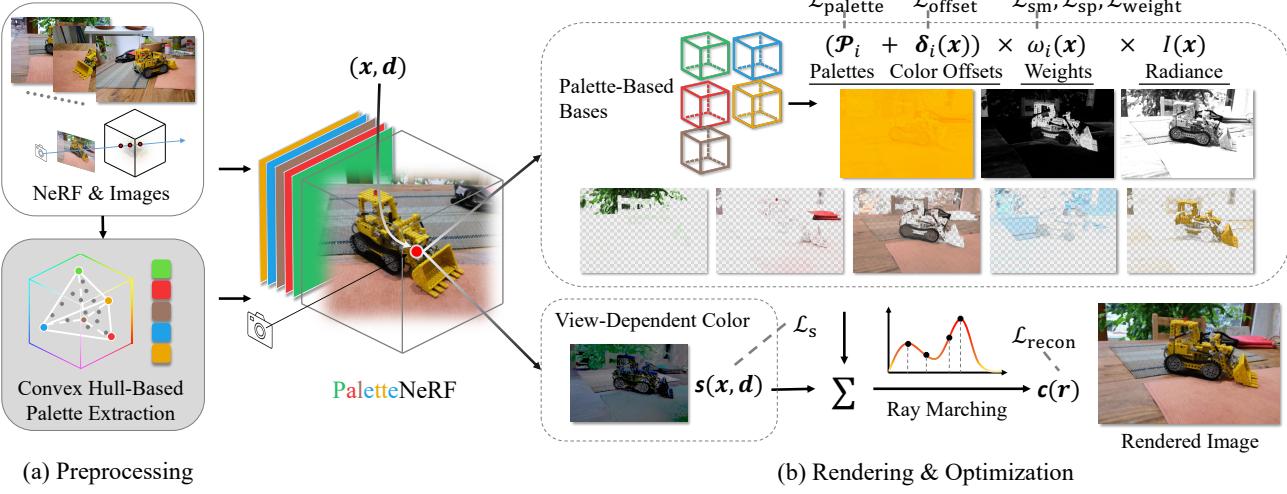


Figure 2. **The overview of our pipeline.** Given a set of training images, we first (a) reconstruct the scene geometry and build the color palettes with existing methods. Then, our **PaletteNeRF** (b) decomposes the scene appearance into multiple palette-based bases and the view-dependent color. We deploy a series of losses on the palette-based base’s functions, the view-dependent color, and the final output.

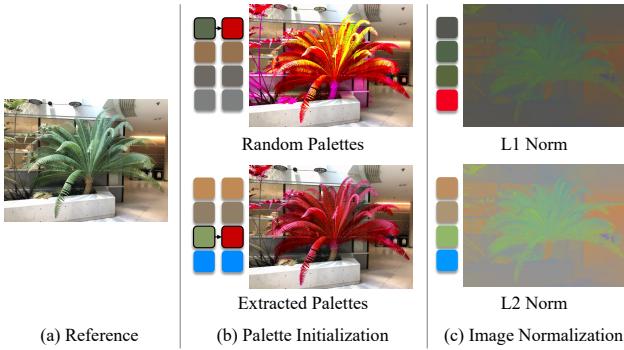


Figure 3. *Left:* Reference image; *Middle:* Comparison between the recoloring result of our model trained with extracted palettes and model trained with random initialized palettes; *Right:* normalized training image and the extracted palettes from normalization of different levels.

$\omega_i(x)$ ,  $\delta_i(x)$  and  $I(x)$ , where  $c_d$  is also fed as an input prior. We compose all of the network outputs to the final color using Eq. 2.

### 3.4. Optimization

Given the fact that separating multiple bases from the scene appearance is a considerably ill-posed task, a lot of cautions should be taken in designing the optimization scheme. Hence, we develop a series of losses to regulate the optimized parameters and to avoid undesirable results such as local-minimum.

To begin with, we deploy the image reconstruction loss

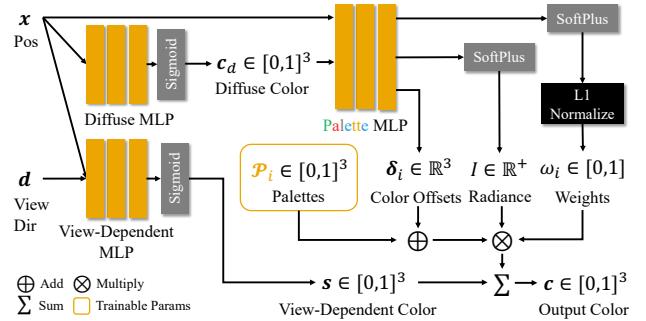


Figure 4. **Our network structure.** As we only show one palette-based basis in the figure, our network generates  $N_p$  bases in parallel and they are summed up in the final step.

defined as:

$$\mathcal{L}_{\text{recon}} = \|c^{\text{ref}} - c(r)\|_2^2 + \|c^{\text{ref}} - (c_d(r) + s(r))\|_2^2, \quad (3)$$

where  $c^{\text{ref}}$  is the ground truth color,  $c(r)$ ,  $c_d(r)$ ,  $s(r)$  are calculated from the volume rendering equation at Eq. 1. The second term of this loss can also be considered as the L2 distance between  $c_d(r)$  and the sum of palette-based bases. We also add a regularization loss  $\mathcal{L}_s$  to the view-dependent color function  $s(x, r)$ , to prevent the particular situation where  $s$  dominates the appearance and pushes all palette-based bases to 0. This loss is defined as:

$$\mathcal{L}_s = \|s(x, d)\|_2^2. \quad (4)$$

While our model uses color offsets to shift the basis color, it is necessary to restrict the blending weights and color offset to avoid extreme solutions. Hence, we adapt

the sparsity loss  $\mathcal{L}_{\text{sp}}$  and the color offset loss  $\mathcal{L}_{\text{offset}}$  from the image soft segmentation method [2]. They are defined as:

$$\mathcal{L}_{\text{sp}} = \frac{\sum_{i=1}^{N_p} \omega_i(\mathbf{x})}{\sum_{i=1}^{N_p} \omega_i^2(\mathbf{x})} - 1, \quad (5)$$

$$\mathcal{L}_{\text{offset}} = \|\delta(\mathbf{x})\|_2^2. \quad (6)$$

The sparsity loss aims to make the blending weights sparser (e.g., segmenting each point  $\mathbf{x}$  to fewer bases), which will eventually increase the capacity of the bases by increasing the color offset. On the other hand, the color offset loss directly suppresses the magnitude of color offsets, preventing them from deviating from the palettes too much. Intuitively, these two losses act as two adversarial roles, and finding a good balance between them will lead to neat segmentation results with reasonable basis color. In our experiments, however, we observe that these two losses may lead to harsh segmentation results, which will drastically affect the quality of the following editing (see our qualitative ablation study). Thus we introduce a novel 3D-aware smooth loss to smooth the weight function based on the NeRF’s output. It is given by:

$$\mathcal{L}_{\text{sm}} = \xi(\mathbf{x}, \mathbf{x} + \varepsilon) \|\omega(\mathbf{x}) - \omega(\mathbf{x} + \varepsilon)\|^2, \quad (7)$$

where  $\boldsymbol{\omega} = \omega_{1 \dots N_p}$ ,  $\varepsilon$  is a random position offset sampled from a Gaussian distribution, and  $\xi(\cdot)$  is the similarity between two points. Here, we adapt the Gaussian kernel used in the bilateral filter, and define the similarity function as:

$$\xi(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma_x} - \frac{\|\mathbf{c}_d(\mathbf{x}) - \mathbf{c}_d(\mathbf{y})\|^2}{\sigma_c}\right), \quad (8)$$

where  $\sigma_x$  and  $\sigma_c$  are smoothing parameters. While the diffuse color  $\mathbf{c}_d$  is used in the smooth loss, we cut off their gradients during the training.

Finally, we add two more losses that incorporate the supervisions from the palette extraction model. They are:

$$\mathcal{L}_{\text{palette}} = \|\mathcal{P} - \bar{\mathcal{P}}\|_2^2, \quad (9)$$

$$\mathcal{L}_{\text{weight}} = \|\omega - \bar{\omega}\|_2^2. \quad (10)$$

As a summary, the overall loss of our optimization is the weighted sum defined as:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{\text{recon}} + \lambda_s \mathcal{L}_s + \lambda_{\text{sp}} \mathcal{L}_{\text{sp}} \\ & + \lambda_{\text{offset}} \mathcal{L}_{\text{offset}} + \lambda_{\text{sm}} \mathcal{L}_{\text{sm}} \\ & + \lambda_{\text{palette}} \mathcal{L}_{\text{palette}} + \lambda_{\text{weight}} \mathcal{L}_{\text{weight}}, \end{aligned} \quad (11)$$

where  $\lambda_s, \lambda_{\text{sp}}, \lambda_{\text{offset}}, \lambda_{\text{sm}}, \lambda_{\text{palette}}, \lambda_{\text{weight}}$  are all loss weights.

### 3.5. Appearance Editing

With the bases predicted from the model, we can simply tune the value of the functions to support appearance

editing, such as recoloring and photorealistic style transfer. When deployed with recent fast NeRF models, our method is able to achieve real-time interactive editing. We will further explain the editing details in the next section.

Since our basis functions are defined on the whole scene, they do not directly support local editing (e.g., edit a single object in the scene). To achieve that, we follow one recent work [17] that learns a 3D feature field from the semantic feature maps predicted from state-of-the-art image-based segmentation models (e.g., Lang-Seg [19]), and use the feature field to guide the editing. However, directly adding high dimensional semantic features to our model may decrease its efficiency, making it impractical to edit in real-time. As the objects captured from a scene are often limited to a small set (e.g., in an indoor scene, chairs, walls and floor are the most likely appeared objects), the semantic features extracted from the scene are often a limited subset of the whole feature space. Therefore, we apply PCA to compress the extracted features to a lower dimension (16 in our experiments) before feeding them to the network.<sup>1</sup>

## 4. Results

### 4.1. Implementation details

**Training.** To support real-time appearance editing, we use Instant-NGP [25] as our backbone. For the geometry learning stage, we keep all original configurations, except adding a per-point rgb loss introduced by Sun et al. [30] to make the density field sparser and avoid floaters. For the segmentation learning stage, we fix the extracted palettes for the first 100 epochs. After that, we unleash the palettes and remove  $\mathcal{L}_{\text{weight}}$  to fine-tune the model. Moreover, since our smooth loss  $\mathcal{L}_{\text{sm}}$  employs the diffuse color  $\mathbf{c}_d$  to calculate the smoothing weight, we do not apply  $\mathcal{L}_{\text{sm}}$  for the first 30 epochs to avoid unconverted  $\mathbf{c}_d$  as input.

Our training and testing experiments are implemented using PyTorch [26], and run on a single NVIDIA RTX 3090 GPU. For both stages, we train our model for 300-600 epochs (depending on the number of training images) using the Adam Optimizer [16] with learning rate set to 0.01, which takes no more than 2 hours in total.

**Datasets.** We conduct experiments on scenes from three sources: Lego, Ficus, Ship and Hotdog from the NeRF Blender dataset [23], Fern, Horns, Flower, Orchids from the forward-facing LLFF dataset [24] and Bonsai, Kitchen and Room from the 360-degree Mip-NeRF360 dataset [3].

### 4.2. Comparisons

**Recoloring.** We use the HSV color space for recoloring in all experiments. Given a group of modified color palettes

<sup>1</sup>Please find more details in terms of implementation, usage and results on semantic guided editing in the supplemental materials.

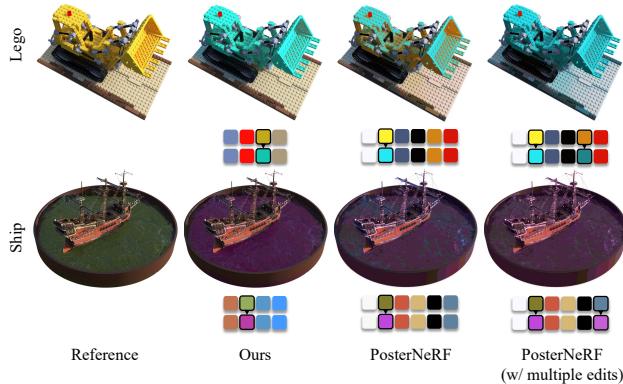


Figure 5. **Qualitative comparison with PosterNeRF [33]**. Since PosterNeRF uses more palettes, we also show their results with edits on multiple palettes for fair comparison. Zoom in for details.

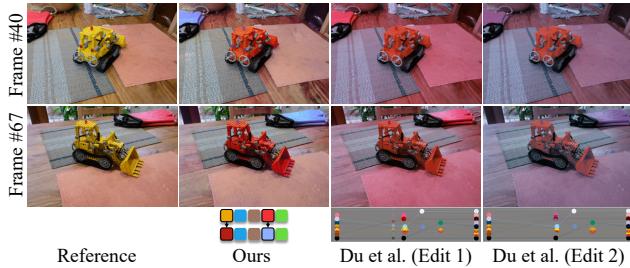


Figure 6. **Qualitative comparison with video palette-based recoloring method [11]**. Unlike Du et al. [11] that either only modifies the lego color (Edit 1) or introduces view-inconsistency when also modifying the gloves (Edit 2), our method recolors the 3D scene consistently while maintaining photorealism across views.

$\mathcal{P}'$ , we calculate the difference between the original palettes  $\mathcal{P}$  and  $\mathcal{P}'$  in HSV, then directly apply the change to all points' soft color, i.e.  $\mathcal{P} + \delta(x)$ .

We compare our model with the state-of-the-art NeRF-based recoloring method PosterNeRF [34] on the Blender Dataset in Fig. 5. Our model produces more photorealistic results with fewer artifacts, and requires fewer palette edits.

We also compare our model with a state-of-the-art palette-based video recoloring model [10] in Fig. 6 and image recoloring models [1, 31] in Fig. 7. Since our model consists of color offsets and radiance functions that extend the palette color to basis, it requires fewer palettes to reconstruct the images than other methods, which improves usability and benefits the quality of recoloring results. Also, with the help of the scene geometry, our results are more plausible in 3D. Particularly, we show recoloring results on two different views in the Kitchen scene, where we set the same recoloring goal for each of them: changing the light on the excavator to purple. While the image-based methods failed to keep the other objects looking the same from the two views, our model is able to produce view-consistent

results for the whole scene.

**Photorealistic style transfer.** We can apply our decomposition model to achieve *photorealistic* style transfer on the captured scenes. Given a style image, users may define a series of correspondences between the image pixels and 3D points, that our system uses to transform our palette-based bases to align the point colors to the pixel colors; more details in the supplementary. We compare our model with three state-of-the-art methods: UPST-NeRF [9], ARF [40] and image-based stylization with AdaIN [14] (Fig. 9). Our results are more photorealistic than ARF and AdaIN, and comparable with UPST-NeRF. However, UPST-NeRF takes much more time to train (tens of hours) than ours.

**User study.** To better demonstrate the effectiveness of our model, we also conduct a user study on these two applications. The study was taken on the Amazon Mechanical Turk platform, where we dispatched 162 questions 30 times to the crowd (4860 in total), and received 4496 answers. In each question, the user watches images/videos generated from our model and another randomly selected baseline and was asked to decide which one is more view-consistent and/or photorealistic. Fig. 8 summarizes our study's results. We outperform image-based recoloring baselines [1, 31], and are considered more photorealistic than posterNeRF [34]. For the style transfer task, our model received better feedback than ARF [40] and AdaIN [14] and is rated better than UPST-NeRF [9] for view-consistency. More details are in the supplementary.

### 4.3. Ablations

We conduct two ablative evaluations on the LLFF dataset to show the effectiveness of our model design. We first compare our model with the original vanilla Instant-NGP with two metrics: the Peak Signal-to-Noise Ratio (PSNR  $\uparrow$ ), the Learned Perceptual Image Patch Similarity [43] (LPIPS  $\downarrow$ ). Our model achieves the same PSNR score from the Instant-NGP (24.50), and has slightly higher LPIPS score (0.152 comparing to 0.145). This shows our model has little effect on the reconstruction quality. We then compare our model with three variants: Model without color offset (Ours w/o  $\delta$ ), model without the sparsity loss (Ours w/o  $\mathcal{L}_{\text{sp}}$ ) and model without the smooth loss (Ours w/o  $\mathcal{L}_{\text{sm}}$ ) both quantitatively and qualitatively. Quantitative results are shown in Tab. 1, where our full model achieves the best sparsity score among all models. Additionally, removing color offset will also decrease the quality of reconstruction in a perceptual way (LPIPS raises from 0.152 to 0.160). We also qualitatively shows comparisons in Fig. 10, where removing smoothness loss leads to major artifacts in recoloring, and removing the sparsity loss results in color bleeding on unrelated areas (walls in the shown case).

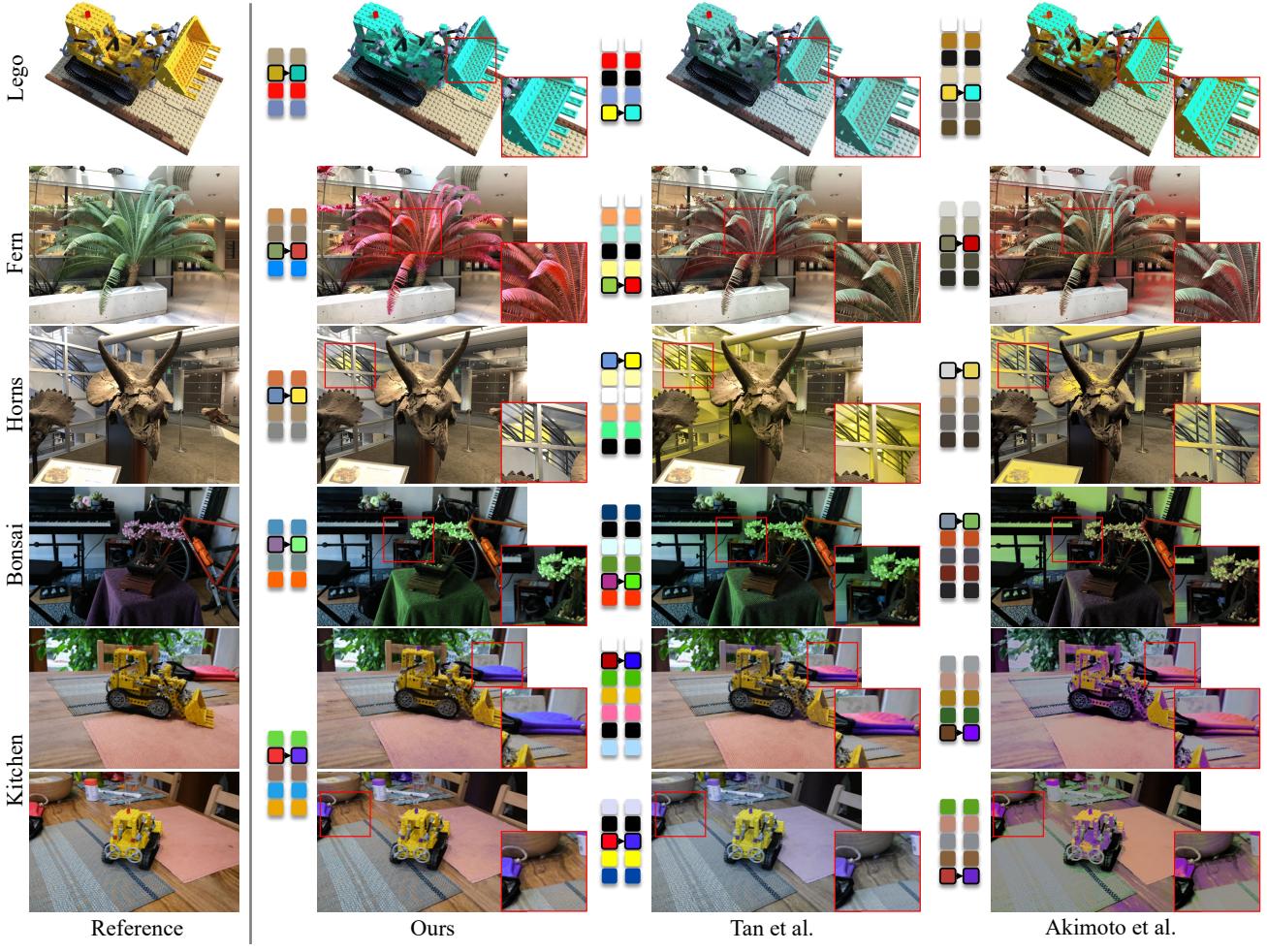


Figure 7. **Qualitative comparison with image palette-based recoloring methods.** We compare our method with Tan et al. [31] and Akimoto et al. [1]. For each recolored image, we also show the corresponding palettes editing on its left side, and a zoom-in view on its right side.

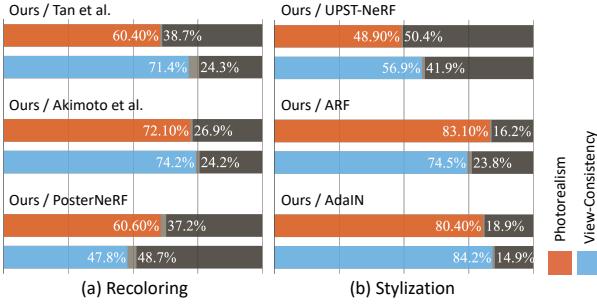


Figure 8. **User study results.** For each comparison we demonstrate the percentage of users who prefer our method, users who prefer the baseline method, and users who do not lean towards anyone.

Table 1. **Quantitative ablation study results.** We compare our model with variants on two metrics: The sparsity error defined by Eq. 5 and the total variation of the weight images (TV) which measures their smoothness.

Methods	Sparsity ↓	TV ↓
Ours	<b>0.478</b>	0.303
Ours w/o $\delta$	0.647	0.263
Ours w/o $\mathcal{L}_{sp}$	1.643	<b>0.123</b>
Ours w/o $\mathcal{L}_{sm}$	0.554	0.549

#### 4.4. More Results

In Fig. 11, we show several results of feature-guided editing. Although our optimized segmentations are shared across the scene and do not directly support local editing, adding semantic features effectively resolves this issue.

In Fig. 12, we show two more edits supported by our decomposition results. By scaling the view-dependent color

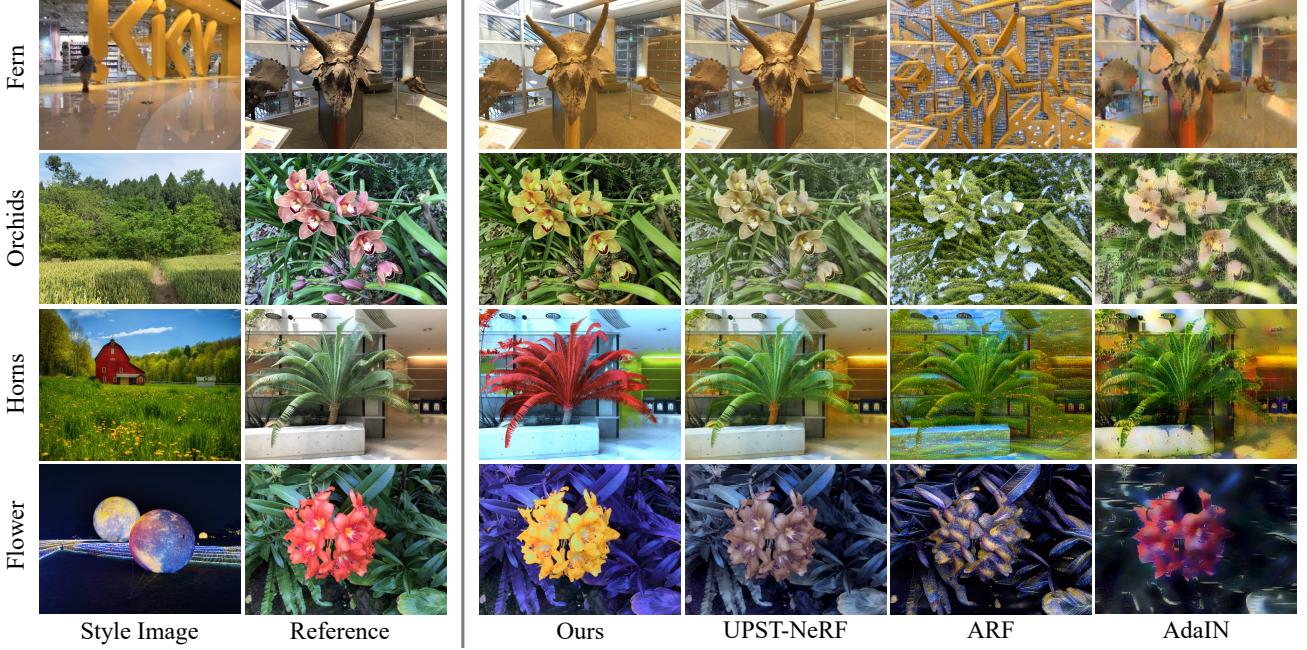


Figure 9. **Qualitative comparison with baseline stylization methods.** Our method produces spatially and temporally consistent style transfer results that faithfully match the style examples while maintaining photorealism.

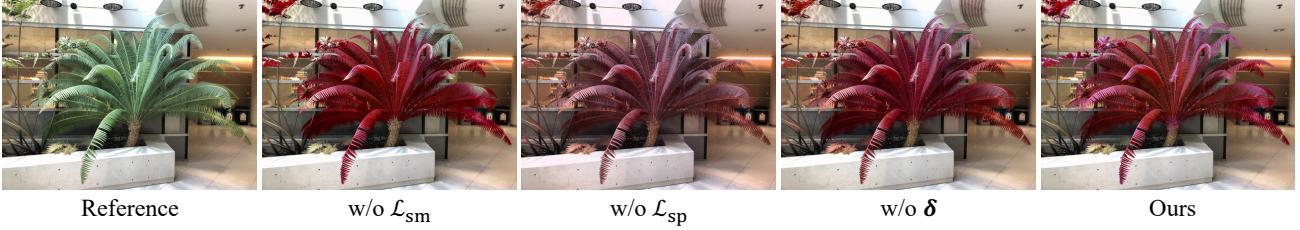


Figure 10. **Qualitative ablation study results.** In this example we change the green palette (primarily representing the plants) to red color.



Figure 11. **Semantic guided editing.** On the left we show the unedited image and recolored results w/o feature guidance. On the right side, we show four guided results where individual objects (plant, floor, table, carpet) are exclusively edited.

function  $s(\mathbf{x}, \mathbf{d})$  and the color offset functions  $\delta_i(\mathbf{x})$ , we can modify the illumination conditions and object textures of the scene, while keeping the rendering photorealistic.

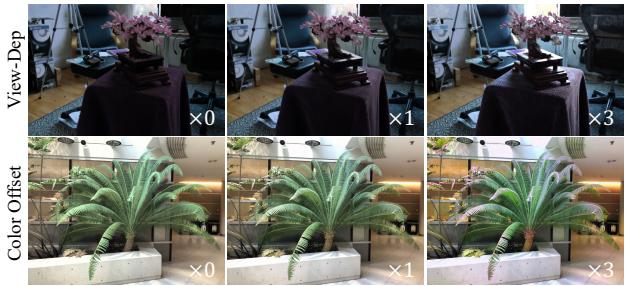


Figure 12. **Additional appearance editing results.** First Row: Rendered images with scaled view-dependent function; Second Row: Rendered images with scaled color offset functions.

## 5. Conclusion

We have presented PaletteNeRF, a novel and efficient palette-based appearance editing framework for neural ra-

diance fields. Our method significantly increases the practicality of palette-based appearance editing and enables intuitive and controllable interactive editing on real-world scenes. Our experiments illustrate the benefit of our approach for various editing tasks such as recoloring, photorealistic style transfer and illuminance changing. Future work may include frequency-based decomposition and editing of specular highlights and extending to dynamic NeRFs.

**Acknowledgement** This project was in part supported by Samsung, the Stanford Institute for Human-Centered AI (HAI), and a PECASE from the ARO. We also thank Jiaman Li for helping conducting our user study.

## References

- [1] Naofumi Akimoto, Huachun Zhu, Yanghua Jin, and Yoshimitsu Aoki. Fast soft color segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8277–8286, 2020. [2](#), [3](#), [6](#), [7](#)
- [2] Yağız Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. Unmixing-based soft color segmentation for image manipulation. *ACM Transactions on Graphics (TOG)*, 36(2):1–19, 2017. [3](#), [5](#)
- [3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. [5](#)
- [4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P. A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12664–12674. IEEE, 2021. [2](#)
- [5] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P.A. Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [2](#)
- [6] Robert Carroll, Ravi Ramamoorthi, and Maneesh Agrawala. Illumination decomposition for material recoloring with consistent interreflections. In *ACM SIGGRAPH 2011 papers*, pages 1–10. 2011. [3](#)
- [7] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):139–1, 2015. [2](#)
- [8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. [1](#), [2](#)
- [9] Yaosen Chen, Qi Yuan, Zhiqiang Li, Yuegen Liu, Wei Wang, Chaoping Xie, Xuming Wen, and Qien Yu. Upst-nerf: Universal photorealistic style transfer of neural radiance fields for 3d scene. *CoRR*, abs/2208.07059, 2022. [2](#), [6](#)
- [10] Zheng-Jun Du, Kai-Xiang Lei, Kun Xu, Jianchao Tan, and Yotam Gingold. Video recoloring via spatial-temporal geometric palettes. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. [2](#), [6](#)
- [11] Zheng-Jun Du, Kai-Xiang Lei, Kun Xu, Jianchao Tan, and Yotam Gingold. Video recoloring via spatial-temporal geometric palettes. *ACM Transactions on Graphics (TOG)*, 40(4), Aug. 2021. [6](#)
- [12] Chie Furusawa, Kazuyuki Hiroshima, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: semi-automatic manga colorization. In Diego Gutierrez and Hui Huang, editors, *SIGGRAPH Asia 2017 Technical Briefs, Bangkok, Thailand, November 27 - 30, 2017*, pages 12:1–12:4. ACM, 2017. [2](#)
- [13] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [2](#)
- [14] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. [6](#)
- [15] Mira Kim, Jaehoon Ko, Kyusun Cho, Junmyeong Choi, Dae-won Choi, and Seungryong Kim. Ae-nerf: Auto-encoding neural radiance fields for 3d-aware object manipulation. *CoRR*, abs/2204.13426, 2022. [2](#)
- [16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [5](#)
- [17] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *CoRR*, abs/2205.15585, 2022. [5](#)
- [18] Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. Neroic: neural rendering of objects from online image collections. *ACM Trans. Graph.*, 41(4):56:1–56:12, 2022. [2](#)
- [19] Boyi Li, Kilian Q. Weinberger, Serge J. Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [5](#)
- [20] José María López-Villegas, Neus Vidal, and Arnau Salas Barenys. 3d-printed broadband power divider based on helical-microstrip transmission line segments. *IEEE Access*, 10:63375–63382, 2022. [2](#)
- [21] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 7210–7219. Computer Vision Foundation / IEEE, 2021. [2](#)
- [22] Abhimitra Meka, Mohammad Shafiei, Michael Zollhoefer, Christian Richardt, and Christian Theobalt. Real-time global illumination decomposition of videos. volume 1, January 2021. [3](#)
- [23] 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. *Commun. ACM*, 65(1):99–106, 2022. [1](#), [2](#), [5](#)

- [24] Pooneh Mohaghegh, Rabia Saeed, François Tièche, Alexis Boegli, and Yves Perriard. Depth camera and electromagnetic field localization system for iot application: High level, lightweight data fusion. In *ASSE 2021: 2nd Asia Service Sciences and Software Engineering Conference, Macau, 24-26 February, 2021*, pages 94–101. ACM, 2021. 5
- [25] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2, 5
- [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5
- [27] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 14315–14325. IEEE, 2021. 1
- [28] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 2
- [29] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 2
- [30] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5449–5459. IEEE, 2022. 5
- [31] Jianchao Tan, Jose Echevarria, and Yotam Gingold. Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. *ACM Transactions on Graphics (TOG)*, 37(6):262:1–262:10, Dec. 2018. 2, 3, 6, 7
- [32] Jianchao Tan, Jyh-Ming Lien, and Yotam I. Gingold. Decomposing images into layers via rgb-space geometry. *ACM Trans. Graph.*, 36(1):7:1–7:14, 2017. 2
- [33] Kenji Tojo and Nobuyuki Umetani. Recolorable posterization of volumetric radiance fields using visibility-weighted palette extraction. *Comput. Graph. Forum*, 41(4):149–160, 2022. 2, 6
- [34] Kenji Tojo and Nobuyuki Umetani. Recolorable posterization of volumetric radiance fields using visibility-weighted palette extraction. *Comput. Graph. Forum*, 41(4):149–160, 2022. 6
- [35] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. *CoRR*, abs/2112.05139, 2021. 2
- [36] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *CoRR*, abs/2201.08845, 2022. 2
- [37] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2
- [38] Weicai Ye, Shuo Chen, Chong Bao, Hujun Bao, Marc Pollefeys, Zhaopeng Cui, and Guofeng Zhang. Intrinsicnerf: Learning intrinsic neural radiance fields for editable novel view synthesis. *arXiv preprint arXiv:2210.00647*, 2022. 2
- [39] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxtels: Radiance fields without neural networks. *CoRR*, abs/2112.05131, 2021. 1, 2
- [40] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields, 2022. 2, 6
- [41] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 5453–5462. Computer Vision Foundation / IEEE, 2021. 2
- [42] Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. Palette-based image recoloring using color decomposition optimization. *IEEE Trans. Image Process.*, 26(4):1952–1964, 2017. 2
- [43] 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
- [44] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. *CVPR*, 2022. 2
- [45] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul E. Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.*, 40(6):237:1–237:18, 2021. 2
- [46] Changqing Zou, Haoran Mo, Chengying Gao, Ruofei Du, and Hongbo Fu. Language-based colorization of scene sketches. *ACM Trans. Graph.*, 38(6):233:1–233:16, 2019. 2