RecolorNeRF: Layer Decomposed Radiance Field for Efficient Color Editing of 3D Scenes

Bingchen Gong*[1], Yuehao Wang*[1], Xiaoguang Han[2] and Qi Dou[1]

[1]*The Chinese University of Hong Kong*
[2]*The Chinese University of Hong Kong (Shenzhen)*

Abstract

Radiance fields have gradually become a main representation of media. Although its appearance editing has been studied, how to achieve view-consistent recoloring in an efficient manner is still under explored. We present RecolorNeRF, a novel user-friendly color editing approach for the neural radiance field. Our key idea is to decompose the scene into a set of pure-colored layers, forming a palette. Thus, color manipulation can be conducted by altering the color components of the palette directly. To support efficient palette-based editing, the color of each layer needs to be as representative as possible. In the end, the problem is formulated as in an optimization formula, where the layers and their blending way are jointly optimized with the NeRF itself. Extensive experiments show that our jointly-optimized layer decomposition can be used against multiple backbones and produce photo-realistic recolored novel-view renderings. We demonstrate that RecolorNeRF outperforms baseline methods both quantitatively and qualitatively for color editing even in complex real-world scenes. https://sites.google.com/view/recolornerf

1. Introduction

Neural radiance fields (NeRF) have been proven to be a powerful representation of 3D scenes, making it likely to become a new media form in the future, similar to images and videos. To this end, supporting the editing of such a new representation is critical. Very recently, several works on this topic have appeared, exploring editing NeRF in terms of geometry deformation [19, 23, 40], appearance editing [11, 20, 39] and style transfer [7, 9, 30, 41] etc. Recoloring, as one kind of appearance editing, usually indicates resetting some specific color tones of a scene for enhancement or correction, which also plays an important role in film production. For an example shown in Fig. 1, by a recoloring edition, the red car can be converted into a blue one in a photo-realistic fashion.

(a) Recoloring Decomposed Layers (b) Rendered Novel Views

Figure 1. Our method recolors an image by automatically decomposing it into layers and recoloring each layer independently. The user can specify the color of each layer, and our method automatically blends the layers.

Among all existing approaches to recoloring an image, palette-based color editing (PCE) [1, 36, 42] provides the most intuitive way for efficient interaction. More specifically, PCE consists of three key steps [42]: 1) Palette extraction. Given a scene, the first step is to determine a set of representative colors and form a palette. 2) Layer decomposition. For each item of the palette, we define a corresponding image layer with a constant color value of the item. Then, the main task of this step is to determine the blending way of these layers to reconstruct the given image. 3) Color editing. Based on the above two steps, the scene

*Both authors contributed equally to this paper. Corresponding email to: bcgong@cse.cuhk.edu.hk

can be recolored intuitively by altering the color of each layer. As one of the SOTA methods of PCE for images, Tan et al. [31–33] proposed a convex-hull simplification strategy for palette extraction. With the obtained palette, layer decomposition is then formulated as an optimization problem. To make the problem solvable, the sparsity of the blending weights is assumed.

In this paper, we propose a novel method, RecolorNeRF, to conduct photo-realistic PCE for NeRF representation, which to our knowledge is the first attempt using a fully learnable palette for layer decomposition. Although PosterNeRF [34] has tried NeRF recoloring based on a palette, it can only enable color editing after posterization, making the results unrealistic. As known, the NeRF of a scene is commonly reconstructed from multi-view images. Thus, another possible way to perform PCE of NeRF is firstly extracting palettes from the pixels in all input images, following the method of [33], and then conducting layer decomposition and color editing for each rendered view of the pre-trained NeRF. Though it is easy to implement, this strategy suffers from three major issues: First, the recoloring in this way becomes post-processing of NeRF rendering, which causes expensive computational costs. Second, as each view is independently processed, the results tend to lack view consistency. Third, the palette extraction is achieved by a heuristic method, which may make the palette color less representative and the layer decomposition not clean enough, therefore interfering with the color manipulation.

To address the above issues, our key idea is to optimize the palette, the layer blending weights, and the volumetric radiance fields in a unified framework. To deal with complicated scenes, we follow [29, 33] to use "over" composition as the resulting image formulation. Specifically, each pixel is represented by alpha blending of a set of ordered layers, where each layer corresponds to an alpha weight. Then, for each layer, we define a volumetric alpha field which can also be modeled by an MLPs similar to the radiance field. Note that, different layers use different MLPs. In this sense, we only need to optimize those MLPs for the blending weights which can also be jointly optimized with the MLPs for the density field. As known, all previous PCE methods conducted palette extraction independently. Our proposed work is the first trial to optimize the palette. To ease the joint optimization problem, two new designs are proposed: 1) To enable a small number of palette colors to accurately represent the whole scene, a novel convex-hull regularization is presented. 2) In order to make the palette color more representative, sparsities on the blending weights are used as usual. To further increase the ability to model complex scenes, a novel order-aware weighting scheme is presented for the sparsity constraint.

Experiments demonstrate that RecolorNeRF can robustly decompose the implicit representation and render photo-realistic images with editable color schemes.

In summary, our main contributions are:

- To our knowledge, we are the first attempt to perform photo-realistic palette-based color editing of NeRF representation using a fully learnable palette for layer decomposition.
- We are the first work to consider jointly optimizing the palette and the alpha blending weights, for which a novel convex-hull regulation is designed to make it solvable.
- The whole RecolorNeRF framework is carefully designed, allowing the color schemes can be efficiently edited even for complicated scenes.

2. Related Work

In this section, we review related work to provide context for our work. We first overview the NeRF editing model variants. Then we review the existing image color editing methods. Finally, we review the methods that extract the palette and decompose the image.

2.1. NeRF Editing

Editing NeRF is challenging due to its implicit representation. Existing works [19, 37, 39, 43, 44] only support editing on local parts of objects or within simple types. NeRF-editing [40] extracts explicit meshes, deforming static objects on the meshes, and transferring the deformations back into the implicit representations. [23] decompose the materials and lighting from images and extract the triangle meshes for 3D reconstruction. HDR-NeRF [11] and RawNeRF [20] use linear raw images as input rather than the post-processed ones, allowing change tone mapping during view synthesis. EditNeRF [19] and CodeNeRF [39] are the first work that capture the shape and color of NeRF on local parts of simple objects as editable latent code. CLIP-NeRF [35] and DFFs [13] support text prompt or exemplar images guided manipulation by leveraging the joint language-image embedding space of the CLIP model. NeRFEditor [30] maps novel-view images to the hidden space of StyleGAN to tune the NeRF with stylization effects. UPST-NeRF [7], INS [9] and ARF [41] use a style transfer network to transfer the style from the reference image to the NeRF model. All of the above adopt additional encoders to extract high-level appearance and shape code, while none of them decomposes the implicit representations in color space and endows intrinsic physical meanings to the decomposition. In contrast, our method can edit both complex scenes and single objects without extra semantic-level features, producing high-fidelity 3D-consistent photo-realistic images. Moreover, our approach supports more controllable

and flexible appearance editing, including precise color tuning and user-designed palettes. During the preparation of this paper, we noticed that there is a concurrent work PaletteNeRF [14] which also supports palette-based editing of NeRF model. Our method is different from theirs in that we tend to decompose the scene into sparser layers using alpha blending and can jointly optimize with randomly initialized palettes, allowing the user to control the composition order of layers against a tailored palette.

2.2. Color Editing

Color editing for images has a long history and has been well-studied in many scenarios. Color editing in images is usually performed by 1) transferring color schemes from the reference image, 2) directly editing the color in a local region, or 3) directly editing the color schemes of an image. Many approaches exploit user guidance by drawing colored scribbles or roughly-drawn spatial markers; such approaches propose to recolor the image by propagating the desired color through the user's brushwork. Algorithms are designed to propagate the user-specified edits to other nearby regions of the image [2, 6, 15, 17, 25, 38]. The problem is often formulated as an energy minimization problem where Euclidean distance or diffusion distance in some feature space is used to evaluate the affinity between pixels. Scribbles impose higher costs on users than reference-based or palette-based methods because the user must provide guidance both in spatial and color. Color transfer is another form of reference-based color editing in which an image takes on another reference image's color. Early color transfer approaches [26, 28] change the colors of the source image based on the color distribution of a reference image. Reinhard et al. [28] perform color transfer by matching the means and standard deviations of the source to reference images. Pitie et al. [26] explore color mapping between source and reference images leveraging probability density functions of colors. Histograms are wildly used to represent the color distribution of images. The color transfer is performed by matching the histograms of the source and reference images. Delon et al. [8] propose a color transfer method that segments the color histogram to construct the palette and warp the source color using the optimal flow from the source palette to the target. While powerful, these color transfer methods provide very few editing controls to users other than the choice of a reference image.

2.3. Palette-based Recoloring

Palette-based methods allow users to specify a set of colors to be used in the recoloring process, providing a trade-off between user control and recoloring. A palette is a succinct representation that can be used to describe the color scheme of an image. Several methods fit predictive models of human palette preference and generate perceptual palettes from input images [3, 10, 16, 24]. A more straightforward approach to extract palettes for color editing is using a k-means method to cluster the existing colors in an image in RGB space [4, 42]. This captures the most prominent colors. A different approach consists of computing and simplifying the convex hull enclosing all the color samples [33], which provides more "primary" palettes that better represent the existing color gamut of the image. Inspired by the geometry structure of the convex hull, we use the convex hull to constrain the color palette of the RecolorNeRF. Once we have the palette extracted, we can use it to represent the colors in the image by decomposing the image into layers, where each layer is a spatially varying importance map of a palette color. Several methods are proposed to composite the layers simply by weighted additive mixing [1, 32, 36, 42]. Though it is simple, the additive mixing is not stacking the layers in order like the Painters' algorithm does. One most common alpha blending operator "over" composite is used in recent methods to composite order-dependent layers [29, 31, 33]. Our method uses alpha blending, and it gives user the control of palette order and produces sparser decomposition.

3. Preliminaries

This section introduces preliminaries of the NeRF model and the color palette-based decomposition, defining the formulation of implicit representation and alpha composition used in RecolorNeRF. Fig.2 shows the overview of RecolorNeRF. We first decompose the NeRF model into layers with the color palette. Then we joint-optimize the layers to recolor the NeRF model. Finally, we render the recolored NeRF model to generate photo-realistic images.

3.1. Neural Radiance Fields

Neural Radiance Fields were introduced by [22] as a differentiable rendering model that can be trained to produce photorealistic images from a single image as input. NeRF is a function that maps a 3D point to a color and a density value. The color and density values are predicted by a neural network. The neural network is trained to minimize the photometric loss between the rendered image and the input image. The NeRF model is a function of the form:

$$F(\mathbf{x}, d) \rightarrow (\mathbf{c}, \sigma) \tag{1}$$

where $\mathbf{x} = (x, y, z)$ is the 3D coordinate of any in-scene point, $d = (\theta, \phi)$ is the viewing distance, $\mathbf{c}$ is the color, and σ is the density. This 5D function is a neural implicit representation (NIR) of any 3D scene. There are also variants of implicit representation defined for high or lower-dimensional spaces. In higher dimensional spatial-time space, $F(\mathbf{x}, d, t)$ could represent a dynamic scene. While in lower-dimensional 2D space, $F(x, y)$ could be

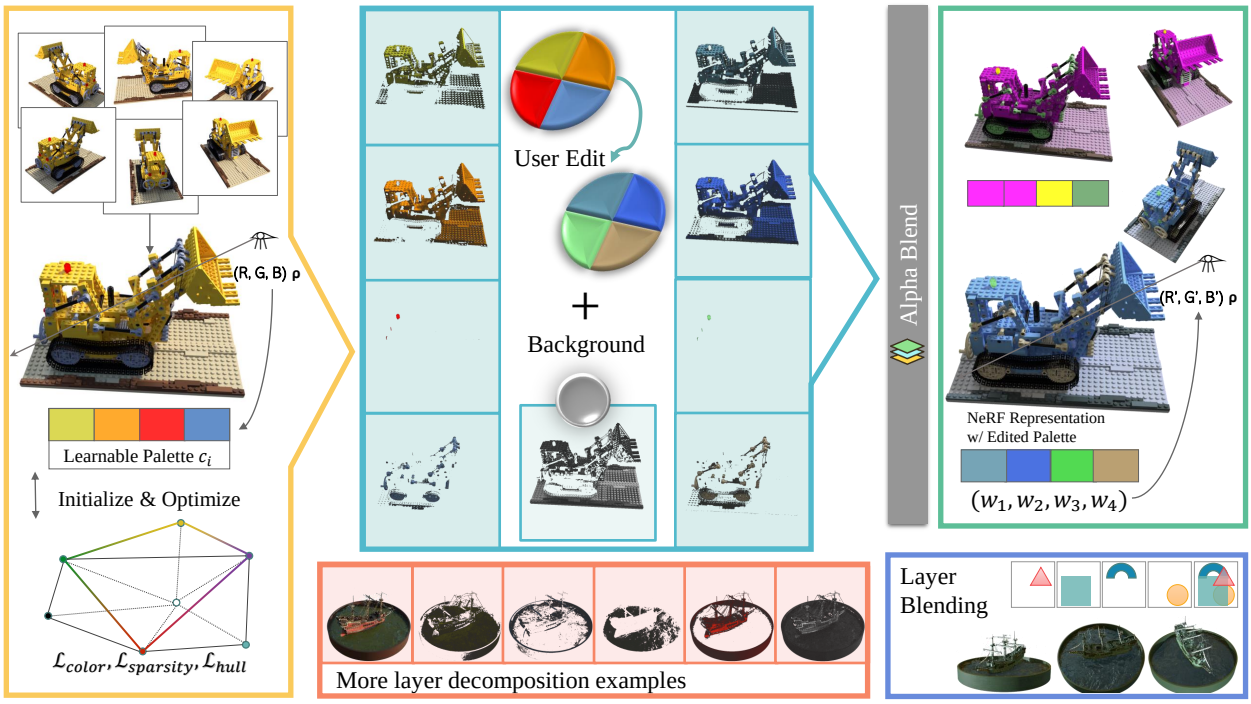

Figure 2. Overview of our method. We first decompose the NeRF model into layers with the color palette. Then we joint-optimize the layers with the palette to fit the layer decomposed radiance field on a specific scene. Finally, we edit the palette and render the recolored NeRF model to generate photo-realistic images. The decomposed layer are stacked together on the user-specified order with alpha blending to support efficient editing on complete scene.

used to compress the image. The implicit representation can be approximated by one or more Multi-Layer Perceptrons (MLPs) sometimes denoted as F_θ. NeRF usually uses volume rendering to produce the image from these colors and densities. The volume rendering is performed by ray marching along the viewing ray. The color and density values are sampled at each step of the ray marching. The color and density values are then combined to produce the final color of the pixel. The volume rendering is differentiable and can be optimized to minimize the photometric loss between the rendered image and the input image.

3.2. Alpha Compositing

Alpha compositing, also called alpha blending, is a technique to combine multiple layers of images into a single image. The layers are composited by blending the layers with the alpha channel. The alpha channel is a grayscale image that controls the transparency of the layer which is usually generated by the user or extracted from the image to blend the layers. The blending operators are first introduced by Porter and Duff [27]. Among many of the commonly used blending operators, the most basic operation of combining two layers is to put one **over** the other as described by Eq. (3), which is, in effect, mimics the normal painting operation (e.g. Painter's algorithm):

$$C_o = C_a\alpha_a + C_b\alpha_b(1-\alpha_a) \tag{2}$$

$$\alpha_o = \alpha_a + \alpha_b(1-\alpha_a) \tag{3}$$

where radiance C_a with translucency α_a is placed over radiance C_b with translucency α_b to produce radiance C_o and translucency α_o. When alpha compositing is in use, each pixel has an additional opacity recorded in its alpha channel and is expressed as a numeric value between 0 and 1. Opacity 0 means that the pixel has no overlay color and is transparent, i.e., the pixel beneath will show through, while 1 means that the pixel is opaque and the geometry completely covers this pixel. Because the paint compositing operation is a linear blend between two paint colors (Eq.3), all pixels in the painting lie in the RGB-space convex hull formed by the original paint colors. Therefore, any pixel color p can be expressed as the convex combination of the original paint colors c_i as follow:

$$p = \sum w_i c_i, w_i \in [0,1] \tag{4}$$

where the w_is are generalized barycentric coordinates rather than the opacity values. The generalized barycentric coordinates do not depend on the layer order and $\sum w_i = 1$.

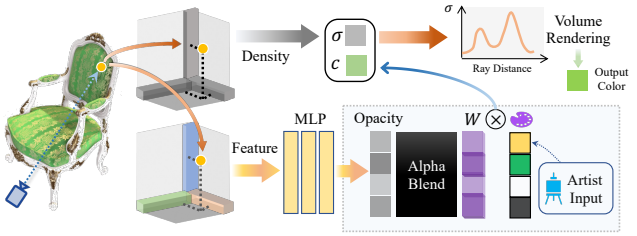

Figure 3. The network architecture of our palette-based automatic decomposition method RecolorNeRF. The layer decomposed radiance field is trained on multiview images. Once optimized, artist input may add to the palette, producing recolored renderings.

They can be converted into layer opacities α_i as follows:

$$\alpha_i = \begin{cases} 1 - \frac{\sum_{j=0}^{i-1} w_j}{\sum_{j=0}^{i} w_j} w_i & \text{if } \sum_{j=0}^{i} w_j \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The conversion from generalized barycentric coordinates to opaque layers relies on a predecided order. Due to the division by zero in the second case, the order is undeterminable when a layer is fully opaque(covering everything underneath). Due to this ambiguity, we propose to predict layer opacities directly from the implicit representation.

4. Jointly-optimized Layer Decomposition

In this section, we describe the RecolorNeRF method, including automatic layer decomposition, alpha blending activation, and the joint-optimization with the color palette. Given the NeRF model and the palette $P = \{c_1, c_2, \cdots, c_K\}$ with K colors, we first decompose the NeRF model into K layers, where i-th layers corresponding to i-th palette. An additional palette c_0 is added to P and acts as the background color. The NeRF model computes view-dependent opacity values $\alpha_i(x, d)$ for each layer i, position x, and view directions d to form implicit fields of layers in addition to the density field.

4.1. Palette-based Automatic Decomposition

Automatic layer decomposition relies on the color palette of the image. The color palette is a set of colors usually extracted from the image and used to represent the image. The key to a successful layer decomposition is a good palette that closely captures the underlying colors the image was (or could be) composite with, even if those colors do not appear in their purest form in the image itself.

Convex Hull Regulation Tan et al. [33] proposed to compute vertices on the convex hull of the pixel colors based on the observation that the color distributions from paintings and natural images take on a convex shape in RGB space. Since the convex hull tightly wraps the observed colors, its vertex colors can be blended with a convex combination to reproduce any color in the image and be used as the palette. They also propose an iterative simplification algorithm to reduce the palette to a user-desired size. After simplification, the remaining colors in the palette are the representative colors in the image suitable to decompose the image into layers. Inspired by the above observation, we use the convex hull to form a feasible region to constrain the choice of colors in the palette of the RecolorNeRF. The convex hull U is then given by the expression:

$$U = \{\sum_{j=1}^{N} \lambda_j c_j | \lambda_j \geq 0 \text{ for all } j \text{ and } \sum_{j=1}^{N} \lambda_j = 1\} \tag{6}$$

where N is the total number of pixels in all multiview images, and c_j is the color of the j-th pixel. The convex hull U is a convex set in the 3-dimensional RGB color space. All colors, including the color of palettes of RecolorNeRF, are constrained to be in the convex hull U for the layer decomposition and alpha blending.

Learnable Palette The existing color editing algorithms [4, 32, 33, 42] usually pre-compute a set of fixed colors as the palette. However, as the palette simplification process is independent of color distribution in the scene, the precompute palette may not be optimal for the layer decomposition on NeRF model. We propose to learn the palette directly with the optimization of implicit representation from multiview images. The palette is modeled as a set of parameters that can be updated with differentiable rendering with color loss while subject to the convex hull regulation and is learned in an end-to-end manner with the optimization of the implicit representation. We represent the palette by a set of K colors $\{c_1, c_2, \cdots, c_K\}$, where K is the number of colors in the palette. The color c_i is a 3-dimensional vector in RGB space. There are infinitely many solutions to the palette without any constraints and the solution becomes unstable. To learn a good palette as previously described, the convex hull regulation applied to the palette is designed as follows:

$$\mathcal{L}_{hull} = \sum_{i=1}^{K} \delta_i \min_{p \in F_{c_i}} \|c_i - p\|_2 \tag{7}$$

where $F : U \rightarrow 2^U$ is a function that finds a set of the closest point on the convex hull U to the color c_i and δ_i is a weight to balance the contribution of each color in the palette. The definition of F is as follows:

$$F_c = \begin{cases} \{HullVertices(U)\} & c \in U \\ \{Nearest(c|s) \text{ for } s \in S(U)\} & \text{otherwise} \end{cases} \tag{8}$$

where $S(U)$ is the set of all simplices(triangles) of the convex hull U and $Nearest(c|s)$ is the closest point on the simplical facet s to the color c. The convex hull regulation is applied to the palette to ensure that the palette is in the convex hull U. In our implementation, we set $\delta_i = 1e2$ for all c_i outside of the hull to ensure the palette color stays in the feasible region. For all c_i inside the hull, we set $\delta_i = 0.1$ to encourage the color in the palette covering all convex hull vertices.

4.2. Alpha Blending Activation

We composite the layers fields to produce the radiance field and compute the final color $C(r)$. This stage takes the opacity fields $\alpha_i(x, d)$ and the palette $P = \{c_0, c_1, \cdots, c_K\}$ as input and outputs the color field $r(x, d)$:

$$r_j = c_K + \sum_{i=1}^{K} \left[(c_{i-1} - c_i) \prod_{j=i}^{K} (1 - \alpha_j) \right] \tag{9}$$

where the background color c_0 is opaque therefore α_0 is set to constant value 1. The cumulative multiplication in Eq.9 makes the optimization of the opacity fields not numerically stable. To avoid this problem, we reformulate the alpha composition in logarithmic space by converting the opacity to generalized barycentric coordinates:

$$w_i = \begin{cases} \sum_{j=i+1}^{K} \log(1 - \alpha_j) & i = 0 \\ \log \alpha_i + \sum_{j=i+1}^{K} \log(1 - \alpha_j) & 0 < i < K \\ \log(1 - \alpha_i) & i = K \end{cases} \tag{10}$$

Generalized barycentric coordinates express any point p inside a polyhedron as a weighted average of the polyhedron's vertices c_i. We can show that $\sum_{i=1}^{K} exp(w_i) = 1$ and $r_j = \sum_{i=1}^{K} exp(w_{ij}) c_i$. This nice convex property eliminates the need for Softmax-like group activation. We use Sigmoid followed by Eq.10 as the output activation of opaque and name it *alpha blending activation*. The alpha blending activation is differentiable and numerically stable compared to cumulative multiplication.

Determining Layer Order The over operator we use to combine the layers, while linear, is not commutative. Therefore the order of layers matters. For K layers, there are $K!$ different orderings so it is not efficient to try and composite with all possible orderings. We assign pixels in multi-view images to their most contributed layer by computing the L2 distance with the corresponding palette color. Then we sort layers by their pixel counts and use the layer with the smallest pixel count as the topmost layer as we find this arrangement usually decomposes sparser opacity. Other criteria such as the total opacity, gradient of opacity, or Laplacian of opacity could also be used based on the specialty of the scene. The user can also directly assign the order according to human preference. Note that the last layer is always a non-transparent layer corresponding to the background color.

4.3. Rendering equation and optimization

We render the composited layers and optimize the NeRF model with color loss. Once the NeRF model is optimized, We can render the recolored NeRF model to generate photo-realistic images. The rendering is differentiable with the following equation:

$$\mathbf{C}(\mathbf{r}) = \sum_{M} \tau_j (1 - exp(-\theta_j \Delta_j)) r_j \tag{11}$$

and τ_j is the accumulated transmittance, representing the probability that the ray travels from t_1 to t_j without being intercepted, given by:

$$\tau_j = exp(-\sum_{t=1}^{j-1} \theta_t \Delta_t) \tag{12}$$

where $\Delta_t = t_{i+1} - t_i$ is the distance between adjacent samples. This function for calculating $\mathbf{C}(\mathbf{r})$ from the set of (w_i, θ_i) values is trivially differentiable for both palette and opacity. The optimization is supervised by the total squared error between the rendered and true pixel colors $\mathbf{C}(\mathbf{r})$ and $\mathbf{C}_{gt}$:

$$\mathcal{L}_{color} = \sum_{x,d} \|\mathbf{C}(\mathbf{r}) - \mathbf{C}_{gt}\|_2^2 \tag{13}$$

Note that r is the ray shot from the camera. The optimization is performed by minimizing the color loss $\mathcal{L}_{color}$ with respect to the NeRF model parameters and the palette P.

Layer Sparsity Regulation The sparsity of opacity fields $\alpha_i(x, d)$ is preferred during layer decomposition. Sparser weights of palettes imply the palette is more representative and the decomposition is more complete. We render the opaque layers to estimate the sparsity of the opacity fields similar to Eq.11:

$$\mathbf{E}_{opqaue}(\mathbf{r}) = \sum_{M} \tau_j (1 - exp(-\theta_j \Delta_j)) \alpha_j \tag{14}$$

and design a specialized soft L0 norm of the rendered opacity fields as the sparsity regulation:

$$\mathcal{L}_{sparsity} = \sum_{x,d} \left\| \frac{1}{1 + \exp\left(-\eta \mathbf{E}_{opqaue}(\mathbf{r}) + 6\right)} \right\|_1 \tag{15}$$

where η is a hyper-parameter standing for the sparsity strength. Since L0 sparsity norm is not differentiable, we alternatively adopt a scaled and shifted sigmoid function to mimic the stepping shape of the L0 norm among $[0, 1]$.

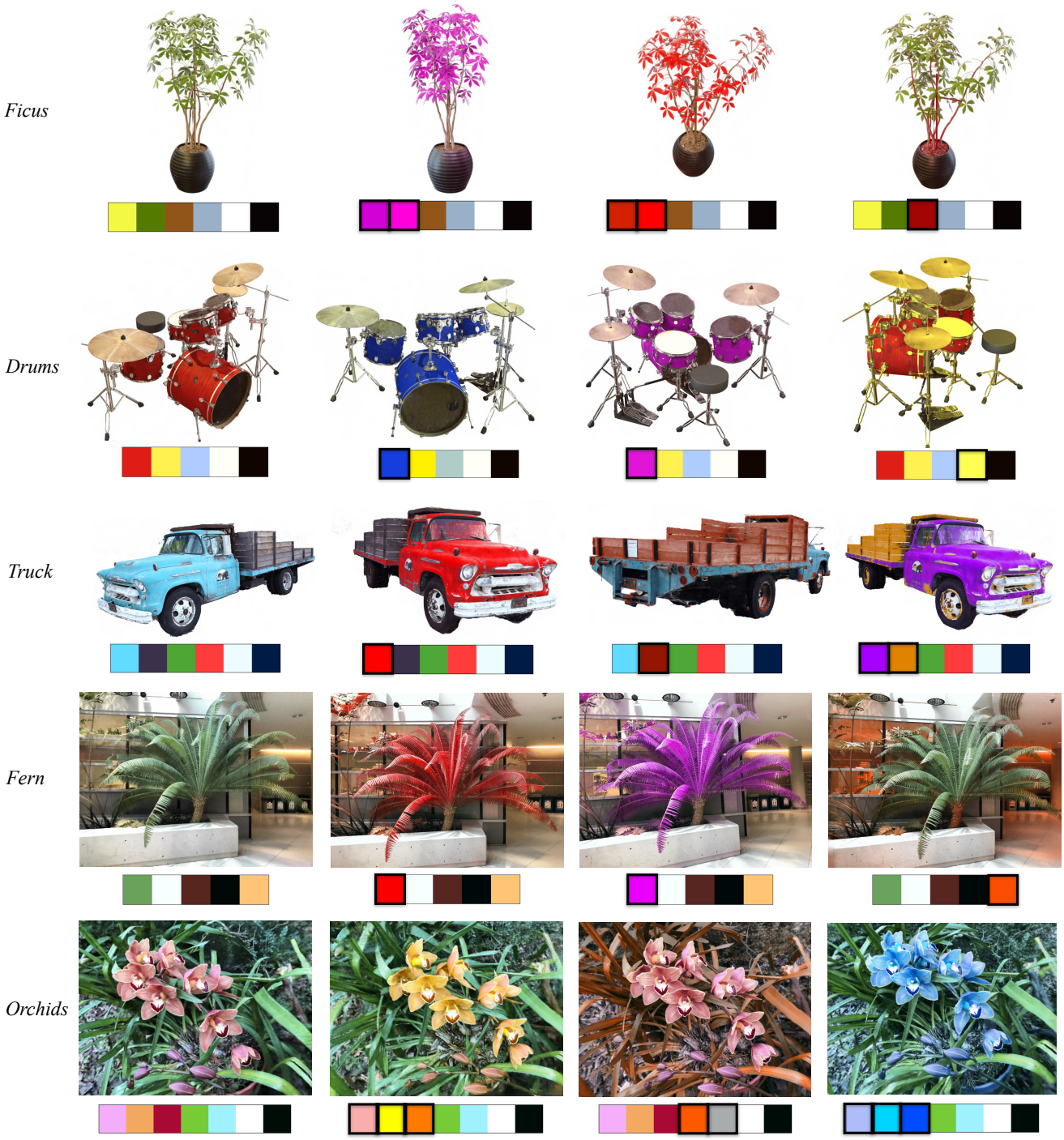

Figure 4. Color editing results of our method. The first image of each row is the reference before editing along with the beneath optimized palette. The other 3 images showcase 3 examples of color editing with the corresponding edited palettes.

Overall Loss We add the regularization to color loss to get the final objective function. The overall loss of our optimization is the weighted sum defined as:

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_{hull}\mathcal{L}_{hull} + \lambda_{sparsity}\mathcal{L}_{sparsity} \quad (16)$$

where λ_{hull} and $\lambda_{sparsity}$ are the hyperparameters for the regularization terms. In our experiments, we set $\lambda_{hull} = 0.1$ and $\lambda_{sparsity} = 0.01$.

5. Experimental Results

In this section, we present our experimental results. We conducted the experiments on the Synthetic NeRF [22], LLFF [21], Synthetic NSVF [18], and Tank and Temples datasets [12,18]. We implemented our model based on TensoRF [5], considering its fast training speed and relatively lower memory cost.

5.1. Qualitative Comparison

We compare our RecolorNeRF with PosterNeRF [34] on the NeRF synthetic dataset. We first set the same editing goal and render novel views for comparison. Figure 5 shows comparison results where we aim to recolor the green chair to a blue chair. In the first edit, our method manages to extract the green components, while PosterNeRF fails to bypass those delicate patterns. In the second edit, the water surface is recolored by our method without polluting the ship's color. In contrast, PosterNeRF blends the colors of the ship and the water surface.

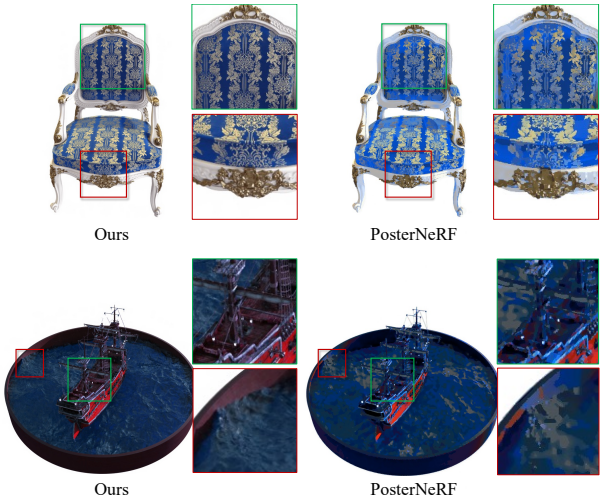

Figure 5. Comparison results of our method and PosterNeRF. In the first edit, we aim to recolor the green chair to a blue chair. In the second edit, we aim to recolor the orange ship to red ship and breen water to ocean blue water

5.2. Ablations

We evaluate the layer decomposition with and without sparsity regulation, alpha blending, and learnable palette. The evaluation is based on the TensoRF backbones. We use the same setting in comparison experiments. The effect of sparsity regulation is shown in Fig. 6 and the effects of alpha blending and learnable palette are shown in Fig. 7.

Sparsity Regularization To evaluate the effect of sparsity regulation, we compare the layer decomposition results of the RecolorNeRF with the sparsity regularization at various strengths ω_{L0}. We can see that without sparsity regularization ($\omega_{L0} = 0$), the decomposed layer are overlapped and the same pixels are associated with multiple layers. With the stronger sparsity regularization ($0.001 \leq \omega_{L0} \leq 0.01$), the decomposed layers are more separated and sparser. The sparsity regularization can also help RecolorNeRF learn a more representative palette by dropping redundant layers and adjusting colors in the palette. If the sparsity regularization is too strong ($\omega_{L0} \geq 0.1$), the sparsity penalty $\mathcal{L}_{sparsity}$ will dominate the color loss $\mathcal{L}_{color}$ and most of the decomposed layers will be dropped. Without enough colors, the reconstruction will fail.

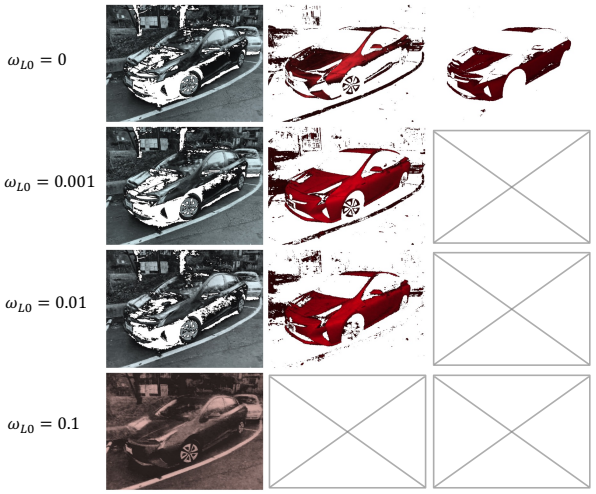

Figure 6. The effect of sparsity regularization on the foreground car color decomposition. The first row shows the decomposition without sparsity regularization. The remaining 3 rows exhibit results under ascending strengths of sparsity regularization.

Alpha Blending with Learnable Palette To evaluate the effect of the alpha blending and learnable palette, we replace the alpha blending with the simple weighting of the palette by opaque and summation to composite the layers (Direct Opaque in Fig. 6). We also fix the palette in the palette in this experiment to evaluate the effect of the learnable palette. We can see that the Direct Opaque scheme fails to effectively decompose the scene into multiple disjoint layers, even though with the regularization of sparsity. The background wall and foreground tree are mixed and distributed into three layers (second, fourth and fifth column). This is due to the direct weighting and summation approach cannot exploit the painting order of layers. Given a reasonable layer order in alpha blending, the background wall and foreground tree are separated into two layers (third and fifth columns). Furthermore, the learnable palette with sparsity regularization can promote the separa-

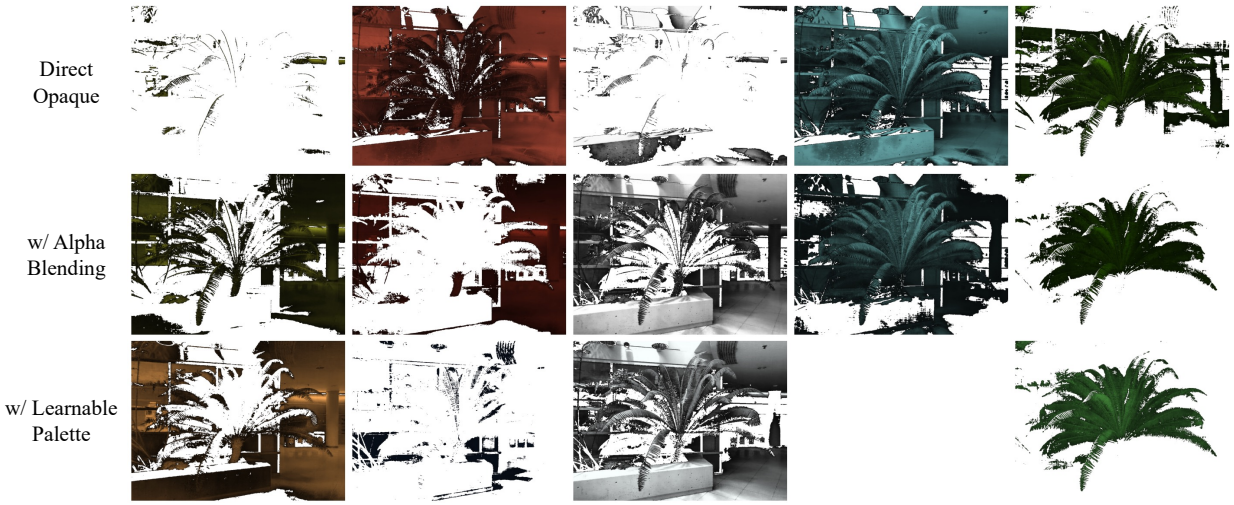

Figure 7. The effect of alpha blending. The first row is the decomposition through predicting opaque directly. The second row shows the decomposition results via alpha blending. The third row further demonstrates the decomposition by combining alpha blending and learnable palettes.

tion of the background and foreground layers by merging the green palettes and dropping the redundant layers. We can see that decomposition by our RecolorNeRF (last row) is the most disjoint and representative, which enables color editing more precise and effective.

5.3. More Visual Results

We demonstrate the RecolorNeRF on both synthetic and real-world datasets in Fig. 4. Our RecolorNeRF can generate photo-realistic free-view images in challenging scenarios. In the first "Ficus" scene, our method can recolor the small twigs to "rosewood" (the last column). Moreover, the specular lighting components are editable by our method, as shown in the last column of the "Drums" scene. The "Truck" scene validates the recoloring performance of our method on a real 360° scene. In addition to recoloring foreground objects, the last column of the "Fern" and the third column of the "Orchids" scenes illustrate the effects of background editing by our method. We include more visualization results in the supplementary material.

6. Conclusion

RecolorNeRF is a simple and effective method to generate recolor photo-realistic images on-the-fly. However, there are still some limitations in the current method. We are the first to propose a method to decompose neural radiance fields into multiple pure-colored layers for recoloring. The decomposed layer are jointly optimized with a learnable palette to produce more disjoint decomposition and more representative colors in the palette. The layers are then stacked with alpha blending to generate color radiance and render the final photo-realistic images. Recoloring a scene is as simple as altering the color in palettes.

Limitations The RecolorNeRF is only suitable for color editing rather than objects or semantic editing. The layer decomposition is entirely based on the color. Therefore two different objects with the same color will be decomposed into one layer. In this case, their color can't be changed separately. By introducing semantic segmentation, we can decompose the scene into multiple layers based on the semantic information this will be the future enhancement of RecolorNeRF.

References

[1] Yağiz 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. 1, 3

[2] Xiaobo An and Fabio Pellacini. Appprop: all-pairs appearance-space edit propagation. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008. 3

[3] Ying Cao, Antoni B Chan, and Rynson WH Lau. Mining probabilistic color palettes for summarizing color use in artwork collections. In *SIGGRAPH Asia 2017 Symposium on Visualization*, pages 1–8, 2017. 3

[4] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Trans. Graph.*, 34(4):139–1, 2015. 3, 5

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 8

[6] Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan. Manifold preserving edit propagation. *ACM Transactions on Graphics (TOG)*, 31(6):1–7, 2012. 3

[7] 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. *arXiv preprint arXiv:2208.07059*, 2022. 1, 2

[8] Julie Delon, Agnes Desolneux, Jose Luis Lisani, and Ana Belen Petro. Automatic color palette. In *IEEE international conference on image processing 2005*, volume 2, pages II–706. IEEE, 2005. 3

[9] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *European Conference on Computer Vision*, pages 636–654. Springer, 2022. 1, 2

[10] Zunlei Feng, Wolong Yuan, Chunli Fu, Jie Lei, and Mingli Song. Finding intrinsic color themes in images with human visual perception. *Neurocomputing*, 273:395–402, 2018. 3

[11] Xin Huang, Qi Zhang, Ying Feng, Hongdong Li, Xuan Wang, and Qing Wang. Hdr-nerf: High dynamic range neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18398–18408, 2022. 1, 2

[12] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 8

[13] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 2

[14] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. Palettenerf: Palette-based appearance editing of neural radiance fields. *arXiv preprint arXiv:2212.10699*, 2022. 3

[15] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004. 3

[16] Sharon Lin and Pat Hanrahan. Modeling how people extract color themes from images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3101–3110, 2013. 3

[17] Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. Interactive local adjustment of tonal values. *ACM Transactions on Graphics (TOG)*, 25(3):646–653, 2006. 3

[18] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 8

[19] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5773–5783, 2021. 1, 2

[20] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul P Srinivasan, and Jonathan T Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16190–16199, 2022. 1, 2

[21] 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. 8

[22] 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. *Communications of the ACM*, 65(1):99–106, 2021. 3, 8

[23] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022. 1, 2

[24] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Color compatibility from large datasets. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011. 3

[25] Fabio Pellacini and Jason Lawrence. Appwand: editing measured materials using appearance-driven optimization. In *ACM SIGGRAPH 2007 papers*, pages 54–es. 2007. 3

[26] Francois Pitie, Anil C Kokaram, and Rozenn Dahyot. N-dimensional probability density function transfer and its application to color transfer. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1434–1439. IEEE, 2005. 3

[27] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, 1984. 4

[28] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001. 3

[29] Christian Richardt, Jorge Lopez-Moreno, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. Vectorising bitmaps into semi-transparent gradient layers. In *Computer Graphics Forum*, volume 33, pages 11–19. Wiley Online Library, 2014. 2, 3

[30] Chunyi Sun, Yanbing Liu, Junlin Han, and Stephen Gould. Nerfeditor: Differentiable style decomposition for full 3d scene editing. *arXiv preprint arXiv:2212.03848*, 2022. 1, 2

[31] Jianchao Tan, Marek Dvorožňák, Daniel Sỳkora, and Yotam Gingold. Decomposing time-lapse paintings into layers. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015. 2, 3

[32] 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):1–10, 2018. 2, 3, 5

[33] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing images into layers via rgb-space geometry. *ACM Transactions on Graphics (TOG)*, 36(1):1–14, 2016. 2, 3, 5

[34] Kenji Tojo and Nobuyuki Umetani. Recolorable posterization of volumetric radiance fields using visibility-weighted palette extraction. In *Computer Graphics Forum*, volume 41, pages 149–160. Wiley Online Library, 2022. 2, 8

[35] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[36] Yili Wang, Yifan Liu, and Kun Xu. An improved geometric approach for palette-based image decomposition and recoloring. In *Computer Graphics Forum*, volume 38, pages 11–22. Wiley Online Library, 2019. 1, 3

[37] Dejia Xu, Peihao Wang, Yifan Jiang, Zhiwen Fan, and Zhangyang Wang. Signal processing for implicit neural representations. In *Advances in Neural Information Processing Systems*, 2022. 2

[38] Kun Xu, Yong Li, Tao Ju, Shi-Min Hu, and Tian-Qiang Liu. Efficient affinity-based edit propagation using kd tree. *ACM Transactions on Graphics (TOG)*, 28(5):1–6, 2009. 3

[39] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. 1, 2

[40] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 1, 2

[41] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 1, 2

[42] Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing*, 26(4):1952–1964, 2017. 1, 3, 5

[43] Chengwei Zheng, Wenbin Lin, and Feng Xu. Editablenerf: Editing topologically varying neural radiance fields by key points. *arXiv preprint arXiv:2212.04247*, 2022. 2

[44] Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. Mofanerf: Morphable facial neural radiance field. In *European Conference on Computer Vision*, pages 268–285. Springer, 2022. 2