IReNe: Instant Recoloring of Neural Radiance Fields

Alessio Mazzucchelli[1,2] Adrian Garcia-Garcia[1] Elena Garces[3,5]
Fernando Rivas-Manzaneque[4,8] Francesc Moreno-Noguer[6] Adrian Penate-Sanchez[7]

[1] Arquimea Research Center [2] Universitat Politècnica de Catalunya [3] Universidad Rey Juan Carlos
[4] Volinga AI [5] SEDDI [6] Institut de Robòtica i Informàtica Industrial, CSIC-UPC
[7]Universidad de las Palmas de Gran Canaria, IUSIANI [8]Universidad Politécnica de Madrid

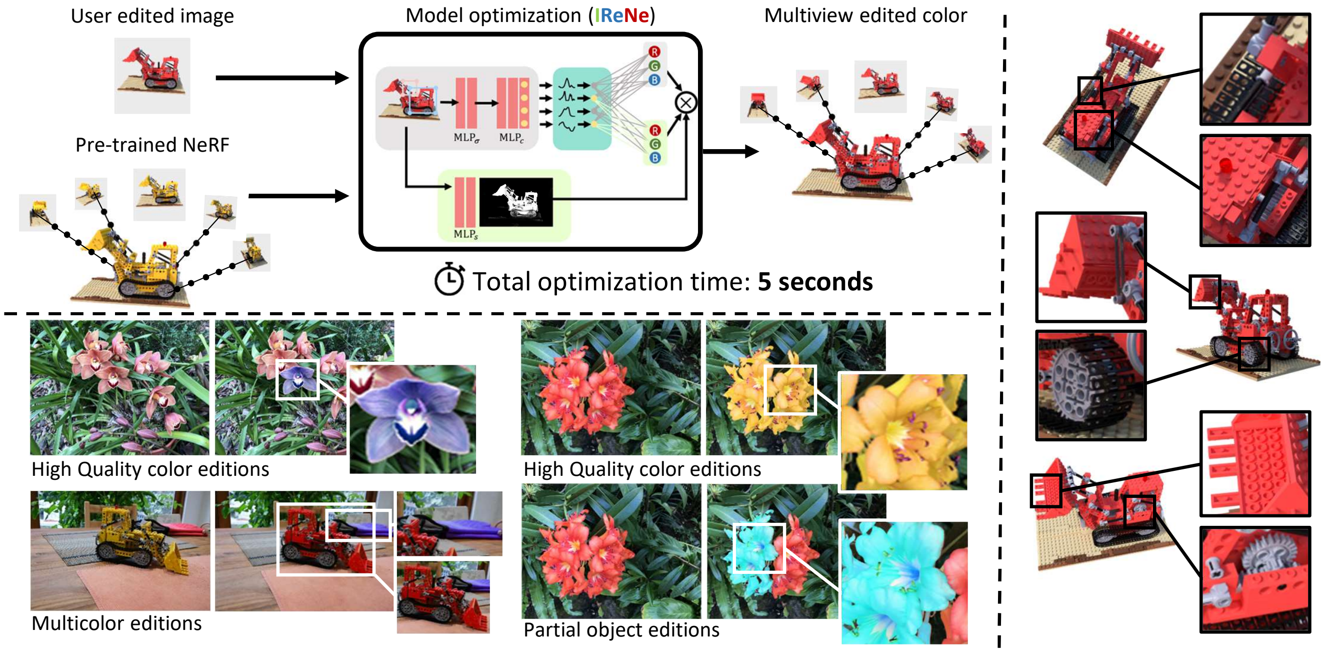

Figure 1. IReNe enables instant 360° recoloring of pre-trained NeRFs using only a single image edited by the user (top row). We introduce an optimization scheme to avoid color bleeding at object boundaries and ensure consistency in view-dependent effects. Furthermore, as illustrated in the bottom row, various types of recoloring are possible, including full-object, partial-object, and multiple-object recoloring.

Abstract

Advances in NERFs have allowed for 3D scene reconstructions and novel view synthesis. Yet, efficiently editing these representations while retaining photorealism is an emerging challenge. Recent methods face three primary limitations: they're slow for interactive use, lack precision at object boundaries, and struggle to ensure multi-view consistency. We introduce IReNe to address these limitations, enabling swift, near real-time color editing in NeRF. Leveraging a pre-trained NeRF model and a single training image with user-applied color edits, IReNe swiftly adjusts network parameters in seconds. This adjustment allows the model to generate new scene views, accurately representing the color changes from the training image while also controlling object boundaries and view-specific effects. Object boundary control is achieved by integrating a trainable segmentation module into the model. The process gains efficiency by retraining only the weights of the last network layer. We observed that neurons in this layer can be classified into those responsible for view-dependent appearance and those contributing to diffuse appearance. We introduce an automated classification approach to identify these neuron types and exclusively fine-tune the weights of the diffuse neurons. This further accelerates training and ensures consistent color edits across different views. A thorough validation on a new dataset, with edited object colors, shows significant quantitative and qualitative advancements over competitors, accelerating speeds by 5× to 500×.

1. Introduction

Neural Radiance Fields (NeRFs) [24] have gained traction due to their ability to construct realistic 3D environments from 2D images and render high-fidelity, photorealistic novel viewpoints. Such advancements unlocked many possibilities, from immersive virtual environments [3, 4, 28] to applications like augmented reality [11, 27]. Nonetheless, the challenge of seamlessly and efficiently editing these neural representations while preserving photorealism remains a critical and largely unexplored frontier. Current NeRF color editing techniques [14, 18, 20] face several limitations that impede their practical applications. First, total required time to perform edition is more than 1 minute for the fastest methods. This makes them ill-suited for interactive applications that demand real-time feedback. Second, current methods often lack accuracy, especially when managing color edits within object boundaries. Lastly, existing editing techniques encounter difficulties in sustaining consistent edits across various viewpoints, especially for scenes intended for 360-degree rendering.

In our paper, we present IReNe, a novel approach that facilitates the Instant Recoloring of Neural radiance fields, effectively addressing the challenges highlighted earlier. Leveraging an off-the-shelf, pre-trained NeRF model (in our case, Instant-NGP [25]) and a single training image featuring user-applied color adjustments to one or various objects, IReNe swiftly fine-tunes the network parameters within seconds. This rapid fine-tuning enables the model to dynamically generate novel scene views, faithfully preserving the color modifications made in the training image.

Our approach is founded on four pivotal contributions. Firstly, we augment the pre-trained NeRF model with a lightweight, trainable segmentation module, providing enhanced control over color edits within object boundaries. Secondly, speed is achieved by selectively fine-tuning the last layer of the network, leveraging only a single training image provided by the user. Thirdly, we show (and take advantage of) that neurons in this last layer show specialisation. Some neurons are responsible for rendering view-dependent appearance while others exclusively contribute to diffuse appearance. We implement an automated classification methodology to distinguish between these neuron types, enabling us to exclusively fine-tune the neurons associated with diffuse appearance. This strategy not only expedites the training process but also guarantees uniform color edits across various viewpoints, enhancing the model's consistency and performance.

Up until now, there has been no dataset available for the quantitative assessment of NeRF color editing methodologies. The fourth contribution of our work involves the development of such a dataset, where the color of certain objects within the NeRF Synthetic [24], LLFF [23] and Mip-NeRF 360 [3] scenes have been meticulously edited manually using Photoshop. Through a comprehensive quantitative and qualitative evaluation on this dataset, it becomes evident that IReNe outperforms [18] by a significant margin. Moreover, it presents considerably better results than [20], addressing issues such as color bleeding outside object boundaries and challenges in maintaining viewpoint consistency. Most notably, our architecture's retraining can be accomplished in under 5 seconds—a stark contrast to the 1-minute requirement of [20] and the 30-120 minutes reconstruction time of [18]. This capability opens the door to seamlessly integrating IReNe into interactive image editing pipelines that demand immediate response. The Dataset is available in our project page.

2. Related Work

Novel view synthesis of 3D scenes is a well-established field. Recently, Neural Radiance Fields (NeRF) [24] marked a breakthrough in this domain. However, editing this new neural representation presents challenges because the 3D information is implicitly encoded in each neuron of a multi-layer perceptron (MLP). Consequently, several works have aimed to extend the original NeRF formulation for different kinds of edits, such as scene relighting [5, 6, 29, 39], scene composition [19, 21, 37], object manipulation [16], inpainting [35], or text-based editing [15, 32–34, 36]. In this work, we focus on the problem of editing the color of the NeRF, addressing the challenges associated with creating predictable, globally coherent, and fast edits.

Color Editing. Editing the color of an image has traditionally been done using two paradigms. Edit-propagation techniques, which involve using local colored cues, such as user scribbles or points, that are propagated to similar areas of the image [1, 2, 10, 13]. The challenges lie in finding a similarity metric that can match the image features of the input exemplar and identify similar points in both local and non-local areas of the image. Deep learning-based approaches have been successful in addressing said challenges, leveraging the effectiveness of convolutional networks as feature extractors [22, 38]. The other paradigm is to use precomputed color palettes [9, 12, 30]. These palettes contain the most relevant colors and serve as a basis for describing the rest through linear compositing. Editing palettes is a choice among artists who might combine it with object segmentation to perform global or local modifications.

NeRF Color Editing. Directly applying color editing methods to a NeRF is not possible as the color values of the scene are implicitly stored in the neurons. A NeRF contains view-dependent effects such as specular reflections that need to be handled appropriately to avoid artifacts. Some of the existing strategies focus on extending the original NeRF architecture so palette-based edits can be applied. RecolorNeRF [14] estimates a color palette in

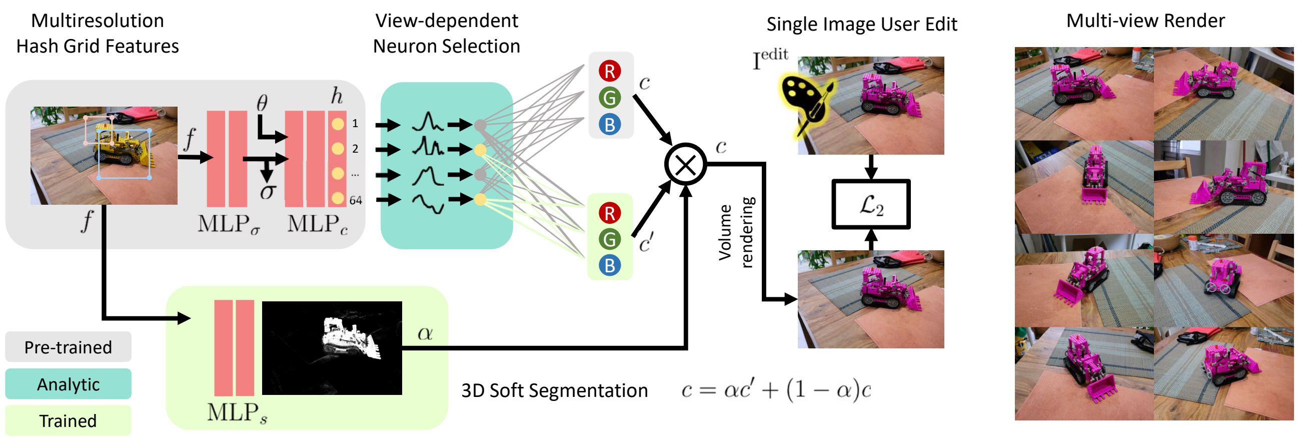

Figure 2. **Overview of IReNe.** We use a pre-trained NeRF and a user-edited training image I^{edit}. Our pre-trained NeRF is an Instant NGP [25] with: a density MLP_σ, with multiresolution hash features f, and a color MLP_c. Mapping the user's edits into the NeRF involves the following steps: 1) Automatic detection of the diffuse neurons in the last layer of MLP_c. 2) Training an MLP_s, ruled by the features f, to estimate a volumetric soft-segmentation α of the edited region. 3) Fine-tuning the weights of the diffuse neurons in the last layer of MLP_c. 4) Alpha blending with the mask α, to estimate the color of a 3D point $\mathbf{x}$ as a linear combination of the color $c_{\mathbf{x}}$ predicted by the frozen weights with the color $c'_{\mathbf{x}}$ predicted by the retrained weights. 5) Volumetric rendering to obtain the edited image I^{render}. MLP_s and the trainable last-layer neurons are optimized through standard RGB loss computation between I^{render} and I^{edit} in under 5 seconds.

pixel space and, through optimization, decomposes each point into a weighted combination of color bases shared across the scene. Palette-NeRF [18] improves it by accounting for specular (or view-dependent) effects. While producing compelling results, this method is very costly, taking hours to train, and is limited to global editions. Although, after said long training, multiple editions can be performed without additional computational cost. Others, such as ICE-NeRF [20], leverage user scribbles to propagate the edition to the full volume encoded by NeRF. In particular, ICE-NeRF utilizes user inputs to apply volume segmentation, which is later used for handling local color edits. As opposed to PaletteNeRF and RecolorNeRF, which optimize the full NeRF, ICE-NeRF fine-tunes only a specific set of neurons of a pre-trained color MLP, achieving faster convergence. However, while fast, its edits suffer from color bleeding at the borders. Furthermore, as acknowledged in the original work, the reprojection method utilized to propagate user inputs is not able to handle complex 360° scenes. Concurrently to us, ProteusNeRF [32], proposes a framework for interactive editing of NeRF, by propagating edits among views based on image features of a pre-trained model [7]. However, their results heavily rely on the view-consistency of these features, which are susceptible to failure under view-dependent effects such as specularities. Also, their edits take between 10 to 70 seconds in contrast to the 5 seconds of our work.

Inspired by these ideas, our method is also retrained from an existing NeRF. However, rather than a complete retraining, we selectively reuse information in the color MLP, akin to ICE-NeRF, expediting convergence. Additionally, we integrate a learnable segmentation branch capable of segmenting editable regions across images, even in sequences with 360° views. As demonstrated in the results section, this segmentation, coupled with a selection and retraining of neurons responsible for diffuse rendering, effectively prevents color bleeding beyond the editable region and guarantees view-consistent edits.

3. Background

Many current NeRF methods use two modules. The first is a set of explicit trainable features that encode each 3D point $\mathbf{x}$ in space in a high-dimensional feature vector $f_{\mathbf{x}}$. The second is a neural network, implemented as two multi-layer perceptrons, MLP_σ and MLP_c. MLP_σ takes as input the feature value for each point and outputs a density value, σ, and a hidden feature vector, h. To estimate the color of the given point, an encoding $\gamma(\cdot)$ [24, 25] is applied to the view direction, θ, that, along with h, is fed into a second MLP_c that outputs the color c of the point $\mathbf{x}$. Each MLP is represented by a set of trainable parameters (ϕ_σ, ϕ_c). Formally,

$$MLP_\sigma(f_{\mathbf{x}}, \phi_\sigma) = \sigma_{\mathbf{x}}, h_{\mathbf{x}} \quad (1)$$

$$MLP_c(h_{\mathbf{x}}, \gamma(\theta), \phi_c) = c_{\mathbf{x},\theta} \quad (2)$$

Our method builds on a pre-trained NeRF using Instant-NGP [25], which implements γ using spherical harmonics. Another key characteristic of this method is that the learnable feature representation $f_{\mathbf{x}}$ is arranged into L levels of a multiresolution hash grid where the number of features per level is determined as a geometric progression. At each level l a hash grid function $f_{\mathbf{x},l}$ produces the value of the

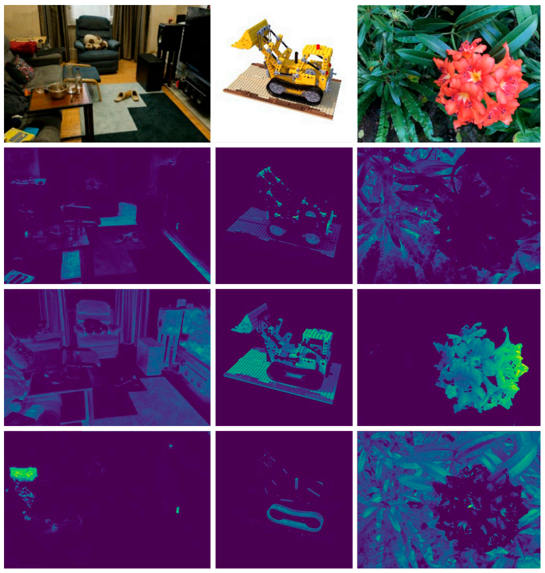
Figure 3. Volumetric rendering for the activations of 3 neurons in the last layer. Points with similar color in 3d space will share a similar activation pattern.

features for the given resolution. The final feature is a concatenation of the features from each level:

$$f_{\mathbf{x}} = \{f_{\mathbf{x},0} \oplus f_{\mathbf{x},1} \oplus ... \oplus f_{\mathbf{x},L}\}. \tag{3}$$

The feature encodings $\{f_{\mathbf{x}}\}$ and the MLP weights for density and color (ϕ_σ and ϕ_c) are learned at the same time in an end-to-end approach by minimizing a standard RGB loss between the volume rendered RGB value for each pixel C_{ij} and the original RGB value from the training image C'_{ij},

$$\mathcal{L} = \sum \|C_{ij} - C'_{ij}\|_2^2 \tag{4}$$

4. Method

Given as input a single re-colored view of the NeRF $\mathrm{I}^{\text{edit}} \in \mathbb{R}^{W\times H\times 3}$, our goal is to instantly propagate this edition to the entire NeRF taking into account the local or global nature of the edit and view-dependent effects. Addressing this requires overcoming several challenges: First, achieving near real-time NeRF editing is crucial. We accomplish this by selectively retraining only the last layer of the color MLP (refer to Sec. 4.1). This approach significantly reduces the number of parameters requiring modification compared to alternative methods [14, 18, 20]. Secondly, since the edition is being performed only on a single view, being able to reproduce coherent view-dependent effects is inherently complex. To solve it, we propose to use the view-dependent information already present in the pre-trained NeRF (Sec. 4.2). Finally, the NeRF edits need to be restricted to the modified regions. We prevent the edits from propagating to undesired areas by introducing a lightweight soft-segmentation network trained on the existing features (Sec. 4.3). Fig. 2 presents an overview of the method.

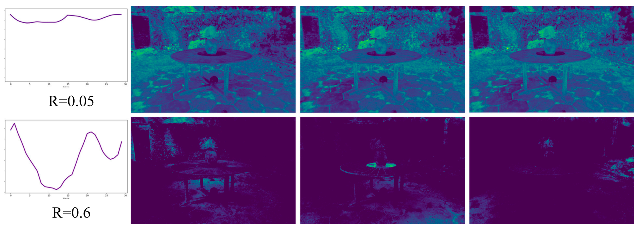

Figure 4. Rendered activations of the same pose while varying the view directional encoding. Top, diffuse neuron. Bottom, view-dependent neuron.

4.1. Last Layer Re-training

Given a pre-trained NeRF model, the color MLP$_c$ can be seen as a mapping ruled by the output hidden vector $h_{\mathbf{x}}$ of the density MLP$_\sigma$, by the view direction encoding $\gamma(\theta)$, and by the parameters ϕ_c of the MLP$_c$, as seen in Eq. (2). A naïve approach to perform the color edition would be to completely retrain the color MLP, obtaining a new set of parameters ϕ'_c for the new set of colors $c'_{x,\theta}$, as follows,

$$\mathrm{MLP}_c(h_{\mathbf{x}}, \gamma(\theta), \phi'_c) = c'_{\mathbf{x},\theta}. \tag{5}$$

However, besides being a costly process due to the full retraining, this also leads to changes of important and useful information already available in the NeRF. Thus, instead of re-training the full MLP, we propose to retrain only the *last layer* of the color MLP$_c$, formally,

$$\mathrm{MLP}_{last}(h_{\mathbf{x},\theta}, \phi'_{last}) = c'_{\mathbf{x},\theta}, \tag{6}$$

where $h_{\mathbf{x},\theta}$ is the output of the penultimate layer, that depends on the position $\mathbf{x}$ and the view direction θ, and ϕ'_{last} is the new set of parameters for the last layer of the color MLP$_c$. Although simple, this strategy enables us to achieve not only rapid edits but also, as demonstrated in Sec. 5.3, to generate higher-quality edits than full re-training. The updated color MLP$_c$ is obtained by exclusively retraining its last layer to match the single image user edit, using Eq. (4). In Fig. 3 we show insights on why our approach performs so well without retraining the whole model. It can be seen that the neuron activations of the last layer show strong evidence of what seems to be color specialisation.

4.2. View-dependent Effects Preservation

The color MLP$_c$ contains all the information necessary to render the color of the scene, including view-dependent effects. Thus, by directly modifying ϕ_{last} using a single image to supervise the training, there is a high risk of modifying this information in an inconsistent manner, generating artifacts. To solve this, we propose a method to reuse the existing view-dependent information in the pre-trained NeRF.

	NeRF Synthetic [24]			LLFF [23]			Mip NeRF 360 [3]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
INGP (ref) [25]	32.88	0.968	0.025	25.95	0.786	0.161	27.32	0.720	0.278
PaletteNeRF [18]	29.92	**0.960**	0.038	24.02	0.763	0.200	22.24	0.666	0.351
RecolorNeRF [14]	29.00	0.943	0.048	24.07	0.714	0.340	-	-	-
IReNe	**30.33**	0.959	**0.035**	**25.63**	**0.783**	**0.165**	**26.80**	**0.714**	**0.292**

Table 1. **Quantitative results and comparison with the state-of-the-art.**

Within the provided h, the 64-dimensional vector coming from the output of the penultimate layer, we identified two distinct behaviors: certain entries in this vector exhibit significant changes as the viewing direction changes, while others maintain constancy, encapsulating what we refer as geometry information (as shown in Fig. 4). We preserve much of the view-dependent information by freezing part of the weights of this last layer. By doing so, pre-existing information is retained after the edition process.

To achieve this, we propose a method to automatically identify entries in h_k exhibiting such behavior. Specifically, for the pixels in the input edited view $\mathrm{I}^{\mathrm{edit}}$, we compute the neuron activations of the last layer, yielding $h_\theta \in \mathbb{R}^{W\times H\times 64}$, for a given view direction θ. We then uniformly sweep θ along the azimuth every ψ degrees, obtaining a set of view-dependent feature vectors $\{h_{\theta_0}, h_{\theta_1}, ..., h_{\theta_Q}\}$ where the value of Q depends on the angular resolution of the scene as we discuss in the implementation details. We aggregate the values of this set along the spatial dimensions obtaining $A \in \mathbb{R}^{64\times Q}$, where each $A_k \in \mathbb{R}^Q$ is a profile of the angular behavior of each output color neuron k across the whole scene (see Fig. 4).

Subsequently, we label the neuron k as a view-dependent neuron if either of the two conditions is met: 1) if $\frac{\mathrm{std}(A_k)}{\mu(A_k)} < \tau_1$, where $\mathrm{std}(\cdot)$ and $\mu(\cdot)$ are the mean and standard deviation, respectively; or 2) if $\min(A_k) \leq \tau_2$. During training, we preserve the weights of neurons identified as view-dependent, thereby retaining their original color. Our strategy of preserving the color of the view-dependent effects is consistent with related methods that explicitly separate view-dependent effects from diffuse reflections using optimization techniques [18].

4.3. Soft 3D Segmentation from 2D User Edits

Since the information encoded on the color MLP_c is global, performing modifications such as the one depicted in Eq. (6) may lead to undesired edits in other regions of the scene. To address this issue, we introduce a 3D segmentation step to selectively apply edits only in the target regions. Since our goal is to achieve near-real-time interactive editing, we avoid relying on 2D segmentation models [8] or user scribbles [20] and the subsequent propagation of segmentation using depth information. Instead, we propose to reuse the already existing information in the feature grid to implement a soft 3D segmentation model for our NeRF. We introduce a small MLP_s that learns a soft-segmentation field $\alpha_{\mathbf{x}} = \mathrm{MLP}_s(f_{\mathbf{x}})$ for the edited area using as input the features f of the multiresolution hash grid. The output of this network is a single value that represents the probability α of a feature at a specific point being part of the edited area.

We use this probability to perform alpha blending between the outputs of the network using the original set of parameters for the color MLP (ϕ_c) and the new ones (ϕ'_{last}):

$$c_{\mathbf{x}} = \alpha_{\mathbf{x}} c'_{\mathbf{x}} + (1-\alpha_{\mathbf{x}})c_{\mathbf{x}}, \tag{7}$$

where $c'_{\mathbf{x}}$ is the new color and $c_{\mathbf{x}}$ is the original one.

The new MLP_s introduced to perform the soft segmentation is trained alongside the last layer of the color MLP_c using the loss presented in Eq. (4), without the need for any ground truth segmentation mask. Convergence is typically achieved in less than 5 seconds.

Implementation details. The soft segmentation MLP_s is implemented as a two-layer MLP. The input layer takes the 32-dimensional multiresolution hash grid feature vector f. The hidden layer has 64 neurons, the output is the one-dimensional probability (α). Selected values of τ_1 and τ_2 are 0.5 and 100 respectively. Q is equal to 30 for all the scenes. For Blender and MIP360 we sweep the whole ψ. Due to its forward-facing nature, only half of the ψ is considered for LLFF. We used the Pytorch [26] implementation of Instant-NGP [25], our models were trained on a single NVIDIA RTX 3090 GPU. Every scene was trained for 200 iterations over the same edited image using the Adam Optimizer [17] with a constant learning rate set to 0.01.

5. Evaluation

In this section we describe how we achieved the first quantitative results of NeRF recoloring methods by creating ground truth edits that allow meaningful and detailed comparisons. We also show qualitative results to visually compare our method with the state of the art.

Dataset. To the best of our knowledge, existing methods lack dedicated metrics to quantitatively evaluate the quality of the edition with respect to ground truth edits. We

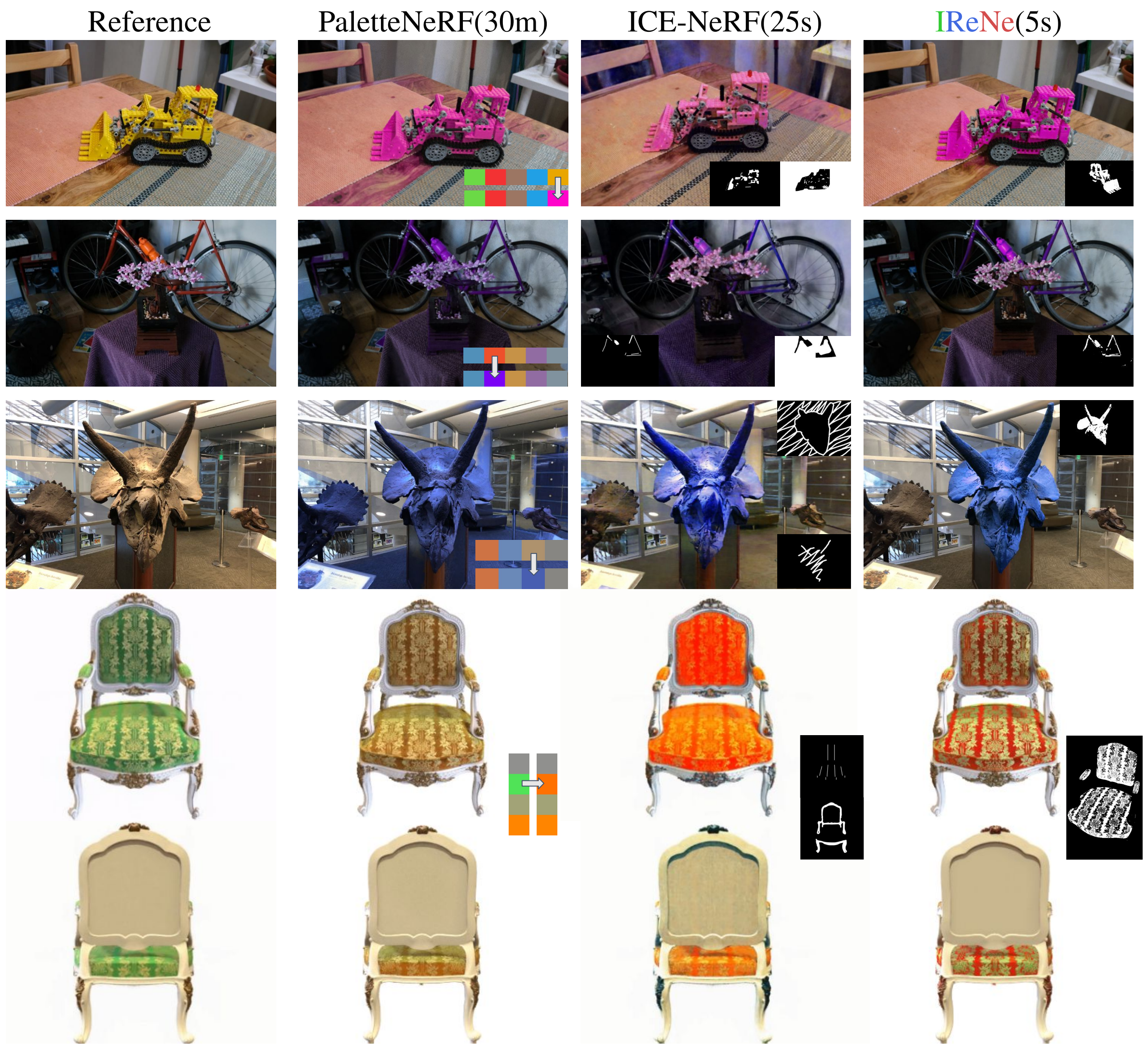

Figure 5. **Qualitative comparison with state of the art methods.** For each approach we show as a small overlayed image the input that the method requires from the user. For PaletteNeRF we show the original and the edited color palettes of the image. For IReNe, we show the region the user selects using Photoshop (or any similar editing tool). On that region we can interactively perform several color edits by modifying the HSV color within the region.

address this issue by creating a new benchmark building on existing datasets. From NeRF Synthetic [24], we used Lego, Mic, Drums, Ficus, and Hotdog (some texture files for Ship, Materials, and Chair are missing), and edited the color map before re-rendering the scene. We then used 1 view for training and 200 for evaluation. From LLFF [23] and MIP360 [3], we used all scenes, for which we apply manually the same edits using Photoshop to 11 views (1 for training, 10 for evaluation).

5.1. Quantitative Evaluation

We quantitatively compare our method with PaletteNeRF [18] and RecolorNeRF [14]. Unfortunately, ICE-NeRF's code is not publicly available, so we just provide qualitative comparisons in the following subsection. Both PaletteNeRF and RecolorNeRF are methods that require a color palette, while our dataset contains full re-colored images as input. Thus, for a fair comparison with these methods, we re-trained the models using our edited image in the RGB loss and froze all the parameters except the palette. In this way, we simulate the process of finding the most appropriate palette in an automated manner (applying stochastic gradient descent). As a result, we obtained the best palette for our edits and were able to emulate our full-image edits.

Quantitative evaluations are reported in Table 1. As both IReNe and PaletteNeRF are trained from Instant-NGP [31], we include metrics from this model on the ten evaluation images (without color edition) as a reference for the maxi-

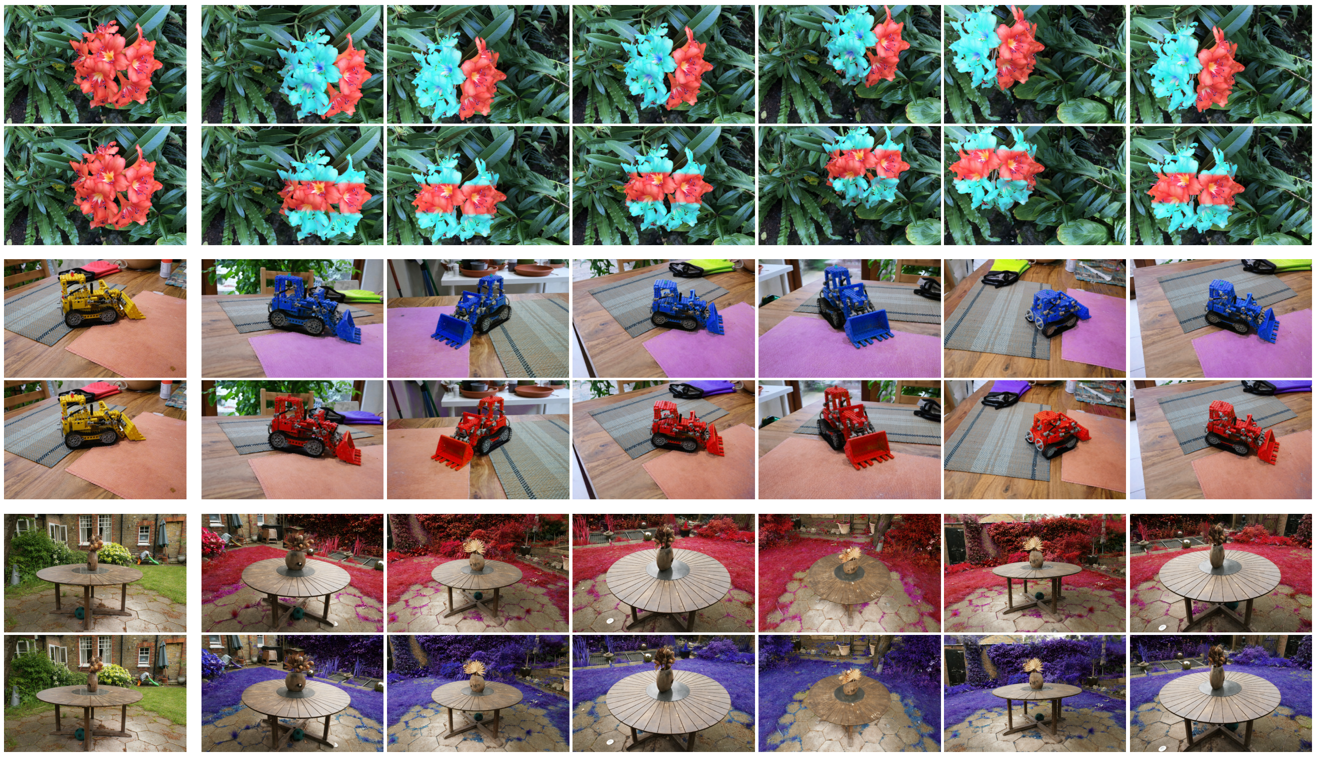
Figure 6. **Additional qualitative results of IReNe.** First column: Original scene, Other columns: Edited scene. **Rows 1-2:** Two partial editions to the flower scene. **Rows 3-4:** Multicolor edition on the same scene. The third row has 3 edits: lego body (yellow to blue), lego emergency light (red to green), oven mitten (red to green). The fourth row has 2 edits. **Rows 5-6:** Single edit to recolor the vegetation.

mum achievable metric. Notably, IReNe outperforms state-of-the-art methods across the three metrics—PSNR, SSIM, and LPIPS. Furthermore, on real datasets (LLFF and Mip NeRF 360), IReNe achieves metrics comparable to Instant-NGP, suggesting proximity to the performance limit imposed by the pre-trained model. Results for RecolorNeRF on the Mip NeRF dataset are not reported due to this method not being able to converge on those scenes. In terms of computation time, our model achieves remarkable speed-ups compared to other approaches. Specifically, the average training time for IReNe is 5 seconds, while RecolorNeRF requires 15-20 minutes, and PaletteNeRF takes between 30 minutes and 2 hours. ICE-NeRF reports training times of 25 seconds. In summary, IReNe achieves between 5× and 500× speed-ups compared to existing approaches.

5.2. Qualitative Evaluation

We offer qualitative comparisons of our method with PaletteNeRF and ICE-NeRF. For PaletteNeRF, we manually selected the palette to achieve the most visually appealing image. Unfortunately, no code was provided for ICE-NeRF, so we had to use the result images from their paper and supplemental material. However, due to the low resolution of ICE-NeRF renders, we chose not to provide zoomed regions in the images to ensure a fair comparison.

Fig. 5 shows the comparison between different methods. Additionally, we present the original image and the supplementary information generated by each method (user scribbles for ICE-NeRF, color palette for PaletteNeRF, and the region selection that the user creates in Photoshop for IReNe before applying the HSV color change in said region). The user edit that IReNe uses is a region selection that the user does on one image, on which we then perform the desired color change in HSV space. For the first 2 results the camera pose of the render was not available for ICE-NeRF so we had to find by hand a similar pose and thus the slight misalignment between the pose used by PaletteNeRf and IReNe and the pose in ICE-NeRFs results.

When analyzing the results in Fig. 5 it can be seen that ICE-NeRF performs poorly in the real scenes from the Mip NeRF 360 dataset, where the color bleeds out into the rest of the scene. Their results are better in Synthetic NeRF dataset but it can be seen that ICE-NeRF destroys part of the contained information when optimizing color. As a way to check if that degradation happens due to the method, the green chair images for the ground truth are obtained from the same ICE-NeRF source and serve as a quality baseline. In the LLFF dataset ICE-NeRF results are more robust. As PaletteNeRF is only able to support global recoloring, the edition is applied outside of the interest region (*e.g.* Horns

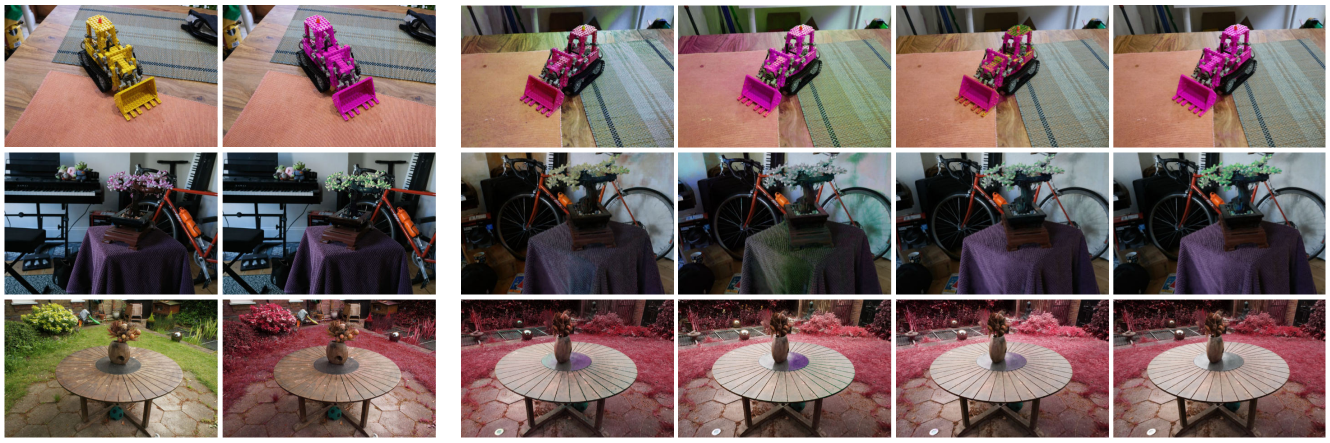
(a) Original (b) Ground Truth Edit (c) Full MLP (d) Last Layer (e) Soft segmentation (f) IReNe

Figure 7. **Qualitative ablation study.** Rendered images from the results detailed in Table 2.

	NeRF Synthetic [24]			LLFF [23]			Mip NeRF 360 [3]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Full MLP	28.25	0.950	0.047	24.86	0.777	0.172	24.31	0.685	0.325
Last layer	29.09	0.951	0.044	24.99	0.779	0.176	25.35	0.702	0.310
Soft Segmentation	30.20	0.956	0.037	24.77	**0.784**	**0.163**	25.68	0.707	0.295
IReNe	**30.33**	**0.959**	**0.035**	**25.63**	0.783	0.165	**26.80**	**0.714**	**0.292**

Table 2. **Ablation study.** In this ablation comparison we show the results of learning the full color MLP instead of just the last layer (Full MLP), the result of only training the last layer but not applying the soft segmentation MLP or the neuron selection (Last layer), the result of using the last layer and the soft segmentation (Soft Segmentation), and the full method IReNe.

scene). It is also spilling some of the color edition into the view-dependent effects of other regions (*e.g.* the pink color for the Lego is propagated to the yellow table reflections). We show additional results of our method in Fig. 6, with multicolor edits and other complex color editions.

5.3. Ablation Study

Tab. 2 provides quantitative results on the quality of different variants of our method. Training uniquely the last layer of MLP$_c$ results in better quality metrics than training the full MLP. Also, adding the soft segmentation further increases the quality of IReNe, while being able to retain existing view-dependent information in the pre-trained model leads to the highest quality. The effect of each of the contributions can be qualitative observed in Fig. 7.

6. Conclusions

We introduced IReNe, a new method that enables interactive NeRF editing with a single user edit. The proposed method not only outperforms current state-of-the-art in terms of speed and interactivity but also overcomes the main limitations of the existing methods, namely multi-view consistency and precision at object boundaries. Besides the high-quality results obtained by IReNe, we also provide insights on how color is encoded in NeRFs. We hope that this work may serve as inspiration to further lines of research, as well as enable use cases and edition workflows that were possible using traditional 2D/3D tools and, now, can be achieved in NeRFs thanks to IReNe.

Limitations and Future Work. One of the main limitations of our work is the need to rely on external edition tools, such as Photoshop, to achieve the complete edition. Also, while the last layer retraining and the method for discriminating which weights to freeze are very robust to the chosen image, the soft segmentation model results may be poor on some occasions. Besides trying to solve these limitations, future work should include the capacity to not only recolor regions but also being able to affect the indirect illumination produced by edited objects onto other objects.

7. Ackowledgements

This work is supported by Arquimea Research Center and Horizon Europe, Teaming for Excellence, under grant agreement No 101059999, project QCircle, and by project MoHuCo (PID2020-120049RB-I00). Elena Garces was partially supported by a Juan de la Cierva - Incorporacion Fellowship (IJC2020-044192-I).

References

[1] Scribbleboost: Adding classification to edge-aware interpolation of local image and video adjustments. In *Computer Graphics Forum*, pages 1255–1264. Wiley Online Library, 2008. 2

[2] Xiaobo An and Fabio Pellacini. Appprop: all-pairs appearance-space edit propagation. *ACM Transactions on Graphics (TOG)*, 27(3):1–9, 2008. 2

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 2, 5, 6, 8

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19697–19705, 2023. 2

[5] 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)*, pages 12664–12674, 2021. 2

[6] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan Barron, and Hendrik PA Lensch. Neural-pil: Neural pre-integrated lighting for reflectance decomposition. In *Advances in Neural Information Processing Systems*, pages 10691–10704, 2021. 2

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, 2021. 3

[8] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. In *NeurIPS*, 2023. 5

[9] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. Palette-based photo recoloring. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. 2

[10] Xiaowu Chen, Dongqing Zou, Qinping Zhao, and Ping Tan. Manifold preserving edit propagation. *ACM Transactions on Graphics (TOG)*, 31(6):1–7, 2012. 2

[11] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16569–16578, 2023. 2

[12] 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), 2021. 2

[13] Yuki Endo, Satoshi Iizuka, Yoshihiro Kanamori, and Jun Mitani. Deepprop: Extracting deep features from a single image for edit propagation. In *Computer Graphics Forum*, pages 189–201. Wiley Online Library, 2016. 2

[14] Bingchen Gong, Yuehao Wang, Xiaoguang Han, and Qi Dou. Recolornerf: Layer decomposed radiance fields for efficient color editing of 3d scenes. *arXiv preprint arXiv:2301.07958*, 2023. 2, 4, 5, 6

[15] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2

[16] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 2

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[18] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. Palettenerf: Palette-based appearance editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20691–20700, 2023. 2, 3, 4, 5, 6

[19] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[20] Jae-Hyeok Lee and Dae-Shik Kim. Ice-nerf: Interactive color editing of nerfs via decomposition-aware weight optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3491–3501, 2023. 2, 3, 4, 5

[21] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 2

[22] Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus Gross. Deep video color propagation. In *Proceedings of the British Machine Vision Conference (BMVC), Newcastle upon Tyne, UK, September 3-6, 2018*. British Machine Vision Association (BMVA), 2018. 2

[23] 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)*, 2019. 2, 5, 6, 8

[24] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2, 3, 5, 6, 8

[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, 2022. 2, 3, 5

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5

[27] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. Web ar: A promising future for mobile augmented reality—state of the art, challenges, and insights. *Proceedings of the IEEE*, 107(4):651–666, 2019. 2

[28] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7491–7500, 2021. 2

[30] 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

[31] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 6

[32] Binglun Wang, Niladri Shekhar Dutt, and Niloy J Mitra. Proteusnerf: Fast lightweight nerf editing using 3d-aware image context. *arXiv preprint arXiv:2310.09965*, 2023. 2, 3

[33] 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 (CVPR)*, pages 3835–3844, 2022.

[34] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2

[35] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. *arXiv preprint arXiv:2305.15094*, 2023. 2

[36] Xiangyu Wang, Jingsen Zhu, Qi Ye, Yuchi Huo, Yunlong Ran, Zhihua Zhong, and Jiming Chen. Seal-3d: Interactive pixel-level editing for neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17683–17693, 2023. 2

[37] 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 *International Conference on Computer Vision (ICCV)*, 2021. 2

[38] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8052–8061, 2019. 2

[39] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.*, 40(6), 2021. 2