LAENeRF: Local Appearance Editing for Neural Radiance Fields

Lukas Radl Michael Steiner Andreas Kurz Markus Steinberger
{lukas.radl, michael.steiner, andreas.kurz, steinberger}@icg.tugraz.at
Graz University of Technology
https://r4dl.github.io/LAENeRF

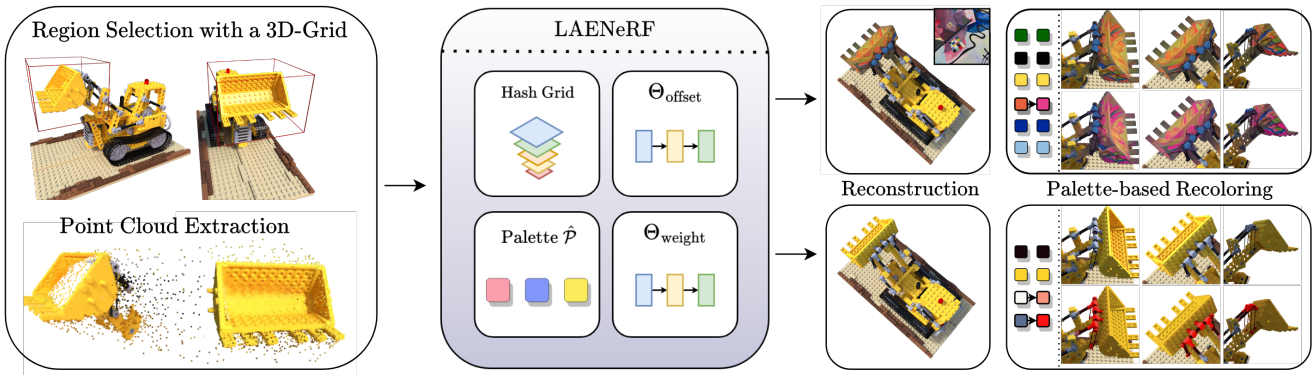

Figure 1. We propose **LAENeRF**, a method for **L**ocal **A**ppearance **E**diting of Neural Radiance Fields. LAENeRF enables appearance edits of arbitrary content in 3D scenes while minimizing background artefacts. For a specified selection, we learn a mapping from estimated ray termination to output colors via a palette-based formulation, which may be supervised by a style loss. In this way, we elegantly combine photorealistic recoloring and non-photorealistic stylization of arbitrary content represented by a radiance field in an interactive framework.

Abstract

Due to the omnipresence of Neural Radiance Fields (NeRFs), the interest towards editable implicit 3D representations has surged over the last years. However, editing implicit or hybrid representations as used for NeRFs is difficult due to the entanglement of appearance and geometry encoded in the model parameters. Despite these challenges, recent research has shown first promising steps towards photorealistic and non-photorealistic appearance edits. The main open issues of related work include limited interactivity, a lack of support for local edits and large memory requirements, rendering them less useful in practice. We address these limitations with LAENeRF, a unified framework for photorealistic and non-photorealistic appearance editing of NeRFs. To tackle local editing, we leverage a voxel grid as starting point for region selection. We learn a mapping from expected ray terminations to final output color, which can optionally be supervised by a style loss, resulting in a framework which can perform photorealistic and non-photorealistic appearance editing of selected regions. Relying on a single point per ray for our mapping, we limit memory requirements and enable fast optimization. To guarantee interactivity, we compose the output color using a set of learned, modifiable base colors, composed with additive layer mixing. Compared to concurrent work, LAENeRF enables recoloring and stylization while keeping processing time low. Furthermore, we demonstrate that our approach surpasses baseline methods both quantitatively and qualitatively.

1. Introduction

Novel view synthesis has been completely revolutionized by Neural Radiance Fields (NeRFs) [28]. NeRFs enable high-fidelity reconstruction of a 3D scene from a set of input images and their camera poses, building on differentiable volume rendering. Recent methods have successfully applied NeRFs to dynamic scenes [34, 56, 57], large-scale scene reconstruction [4, 21, 39] and varying lighting conditions [6, 26]. Local appearance editing of these learned 3D scene representations remains relatively underexplored. The implicit representation used in NeRFs in the form of a multi-layer perceptron (MLP) is the main challenge, causing non-local effects when a single parameter is modified, Distilled feature fields [19] and per-image 2D masks [23] have been suggested as mechanisms to facilitate local edits for NeRFs. However, both of these methods frequently

introduce artefacts in the non-edited regions. Other editing approaches support recoloring a NeRF by remapping individual colors [11, 20, 49], try to extract modifiable material quantities for re-rendering [5, 50], or apply style transfer [13, 54]. Virtually all approaches in these domains do not support controllable local edits, i.e., they always also introduce global changes, which constrains their viability to the theoretical domain and impedes their applicability in practical contexts. At the same time, most approaches struggle with high memory requirements and long compute times, further hampering their use. Ultimately, there is currently no method that enables simultaneous style transfer and interactive recoloring.

To address the previously discussed limitations, we propose LAENeRF, a method for local appearance editing of pre-trained NeRFs. Choosing NeRFShop [16] and Instant-NGP (iNGP) [29] as building blocks, we use a 3-dimensional grid, a subset of iNGP's occupancy grid, as our primitive for selecting scene content. Due to the region growing procedure inherited from NeRFShop, which relies on a growing queue storing direct neighbors, we can model smooth transitions to content adjacent to our selection, resulting in more visually appealing edits. Inspired by previous recoloring approaches [11, 20], we introduce a novel NeRF-like module designed to learn a palette-based decomposition of colors within a selected region. In contrast to previous work, we estimate a per-ray termination point resulting in a point cloud which represents the editable region. This design decision reduces memory requirements and increases performance drastically. We feed these points into our neural LAENeRF module, which learns a palette-based decomposition by jointly optimizing two MLPs and a set of base colors to reconstruct the selected region (see Fig. 1). As LAENeRF learns a function in 3D space, we can implicitly ensure multi-view consistency and prune outliers. The learned set of colors may be modified after optimization to enable interactive recoloring.

By providing a style loss during reconstruction, LAENeRF can stylize the selected region, while keeping its recoloring abilities and processing time low. We propose several novel losses to generate high-fidelity results while respecting the learned 3D geometry and extracting an intuitive color decomposition. Finally, we generate a modified training dataset by blending our edited region with the original training dataset and fine-tune the pre-trained NeRF. Our experiments demonstrate that LAENeRF is not only the first interactive approach for NeRF appearance editing, but also qualitatively and quantitatively outperforms previous methods for local recoloring and stylization.

In summary, we make the following contributions:

(1) We combine photorealistic and non-photorealistic appearance edits for NeRFs into a unified framework.
(2) We propose the first interactive approach for local, recolorable stylization of arbitrary regions in NeRFs.
(3) We propose a new architecture and novel regularizers for efficient, geometry-aware 3D stylization.

2. Related Work

Photorealistic Appearance Editing i.e. recoloring for NeRFs, modifies the underlying material colors, without changing textures or lighting. For this task, several methods [11, 20, 43, 49] learn a decomposition into a set of base colors with barycentric weights and per-pixel offsets with additive layer mixing [37, 38]. This color palette can be modified interactively during inference. Orthogonally, several approaches recover the material properties directly [5, 41, 44, 45, 50, 53]. To further enable local recoloring, feature fields can be jointly optimized during training [19]. PaletteNeRF [20] incorporates this idea, allowing end users to guide recoloring given the selection of a single reference point. However, this approach limits the end user when performing local edits and is prone to introducing artefacts. ICE-NeRF [23] performs local recoloring by modifying the most significant weights in the color MLP of a trained NeRF, given per-image annotations of foreground and background. This approach works well for bounded or forward-facing scenes, but struggles for unbounded, 360° captures due to the large 3D space. In comparison to all previous methods, our approach ensures that edits remain local and provides an intuitive interface to artists.

Non-Photorealistic Appearance Editing for NeRFs modifies textures as well as material properties. Leveraging image style transfer [9, 10, 14], recent methods apply perceptual losses [9, 17] to radiance fields. Among these, several propose separate modules for color and style [7, 13, 15, 32] or progressively stylize a trained NeRF scene [31, 46, 51, 55]. Another line of works utilizes two separate NeRFs, one for reconstruction and one for style [2, 12, 47]. Recent methods have also investigated modifiable stylization: Pang *et al*. [32] can generate different versions of the same style by utilizing a hash grid [29] with modifiable hash coefficients. StyleRF [25] utilizes a feature field for global, zero-shot stylization. Ref-NPR [54] faithfully propagates the style of a reference image to a pre-trained NeRF. Our approach requires significantly less memory and compute resources, leads to higher multi-view-consistency, allows to apply style transfer only locally, and enables recoloring of the stylized radiance field within an interactive framework.

3. Preliminaries

In this section, we revisit volumetric rendering with NeRFs and outline our procedure for region selection.

Neural Radiance Fields. NeRFs [28] learn a function

$$\Theta_{\text{NeRF}} : \mathbb{R}^5 \to \mathbb{R}^4,\ (\boldsymbol{p}, \boldsymbol{d}) \mapsto (\boldsymbol{c}, \sigma), \tag{1}$$

where $\boldsymbol{p} = (x, y, z)$ and $\boldsymbol{d} = (\theta, \phi)$ denote the sample position and viewing direction, $\boldsymbol{c} \in [0, 1]^3$ denotes the predicted output color and $\sigma \in \mathbb{R}_+$ denotes the predicted volumetric density. For each pixel, positions $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ along a ray from the camera position $\boldsymbol{o}$ in the direction $\boldsymbol{d}$ are sampled. At N points $t_i : 0 < i \leq N$ along $\boldsymbol{r}$ between the near and far plane, the colors and density are evaluated and composed using volumetric rendering:

$$\hat{\mathcal{C}}(\boldsymbol{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\boldsymbol{c}_i, \tag{2}$$

where $\delta_i = t_{i+1} - t_i$ and the transmittance T_i is given as

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right). \tag{3}$$

Through this fully differentiable pipeline, Θ_{NeRF} can be optimized with $\sum_{\boldsymbol{r}\in\mathcal{R}} \|\hat{\mathcal{C}}(\boldsymbol{r}) - \mathcal{C}(\boldsymbol{r})\|_2^2$ for a set of rays $\mathcal{R}$, where $\mathcal{C}(\boldsymbol{r})$ denotes the ground truth color. With a trained Θ_{NeRF}, we can compute the estimated depth ζ using

$$\zeta = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))t_{i+1}, \tag{4}$$

which can in turn be used to compute the estimated ray termination

$$\boldsymbol{x}_{\text{term}} = \boldsymbol{o} + \zeta\boldsymbol{d}. \tag{5}$$

Region Selection with NeRFShop. NeRFShop [16] leverages a voxel grid, akin to the occupancy grid $\boldsymbol{\mathcal{O}}$ of iNGP [29], to select arbitrary content in iNGP-based radiance fields. The occupancy grid of iNGP is an acceleration structure, discretizing the bounded domain into uniformly sized voxels, with a value of 0 or 1 depending on the density. We utilize this concept with our edit grid $\boldsymbol{\mathcal{E}}$. To facilitate region selection, the user scribbles on the projection of the 3D scene on the screen. Subsequently, for each selected pixel, a ray is cast and the estimated ray termination $\boldsymbol{x}_{\text{term}}$ is computed with Eq. (5). Next, we map each $\boldsymbol{x}_{\text{term}}$ to the nearest voxel in our edit grid $\boldsymbol{\mathcal{E}}$ and set the corresponding bit in the underlying bitfield. For intuitive selection, we support region growing from the selected voxels, by adding the neighboring voxels to a growing queue $\mathcal{G}$. During region growing, we add the current voxel to $\boldsymbol{\mathcal{E}}$ and its neighbors to $\mathcal{G}$ if the occupancy grid is set. Through this workflow, we offer an intuitive method for content selection within iNGP's hybrid representation.

4. LAENeRF

Our key insight is that we can reduce memory and computational requirements by optimizing a small, NeRF-like network given the estimated ray termination $\boldsymbol{x}_{\text{term}}$. Given the implicit assumption that our pre-trained NeRF accurately reconstructs the scene geometry, our formulation is inherently view-consistent. We present our network architecture in Fig. 2. $\boldsymbol{x}_{\text{term}}$ is featurized using a trainable multi-resolution hash grid [29]. The encoded input $\xi(\boldsymbol{x}_{\text{term}})$ is subsequently passed to shallow MLPs to predict per-ray barycentric weights $\hat{\boldsymbol{w}} \in [0, 1]^{N_{\hat{\boldsymbol{\mathcal{P}}}}}$ and offsets $\hat{\boldsymbol{\delta}} \in [-1, 1]^3$, where $N_{\hat{\boldsymbol{\mathcal{P}}}}$ denotes the size of the learnable color palette $\hat{\boldsymbol{\mathcal{P}}} \in \mathbb{R}^{N_{\hat{\boldsymbol{\mathcal{P}}}}\times 3}$. With our predicted intermediate values, we compose our output color as

$$\hat{\boldsymbol{c}} = \text{clamp}\left(\hat{\boldsymbol{w}}^T\hat{\boldsymbol{\mathcal{P}}} + \hat{\boldsymbol{\delta}}\right). \tag{6}$$

To allow the offsets to model view-dependent effects, we featurize $\boldsymbol{d}$ using a spherical harmonics encoding [8, 44] and use this as an additional input to our offset network.

Input: Estimated Ray Termination. As can be seen in Fig. 2, LAENeRF learns a mapping from $\boldsymbol{x}_{\text{term}}$ to estimated output color $\hat{\boldsymbol{c}}$. However, we only require $\boldsymbol{x}_{\text{term}}$ of rays $\boldsymbol{r}$ which intersect $\boldsymbol{\mathcal{E}}$. To correctly handle occlusions and mitigate errors due to the edit grid resolution, we require the alpha accumulated inside the edit grid to be larger than a certain threshold, $\tau_{\text{edit}} = 0.5$. To obtain accurate depth estimations, requiring the full accumulation of alpha, we simultaneously perform raymarching through both $\boldsymbol{\mathcal{E}}$ and the occupancy grid $\boldsymbol{\mathcal{O}}$, computing ζ within the occupancy grid.

Style and Content Losses. As is common in 2D image style transfer [9, 17], we use separate losses for content and style, in our case:

$$\mathcal{L}_{\text{content}} = \|\hat{\boldsymbol{c}} - \boldsymbol{c}\|_2^2, \tag{7}$$

$$\mathcal{L}_{\text{style}} = \lambda_{\text{style}}\|\boldsymbol{G}(\boldsymbol{s}) - \boldsymbol{G}(\hat{\boldsymbol{c}})\|_2^2, \tag{8}$$

where $\boldsymbol{s}$ denotes an arbitrary style image and $\boldsymbol{G}(\cdot)$ denotes the gram matrix [9] of a feature extracted from a semantic encoder, *e.g.* VGG19 [36]. Crucially, we can perform photorealistic recoloring by setting $\lambda_{\text{style}} = 0$, where LAENeRF learns to reconstruct the selected region.

Geometry-preserving Losses for Stylization. We notice that when we perform non-photorealistic stylization of 3D regions using only $\mathcal{L}_{\text{content}}$ and $\mathcal{L}_{\text{style}}$, small structures are often eliminated in favor of a consistent stylization. In addition, as imperfect geometry reconstruction from our pre-trained Θ_{NeRF} leads to noise in $\boldsymbol{x}_{\text{term}}$, we need additional regularization to encourage smooth, low-noise outputs. To

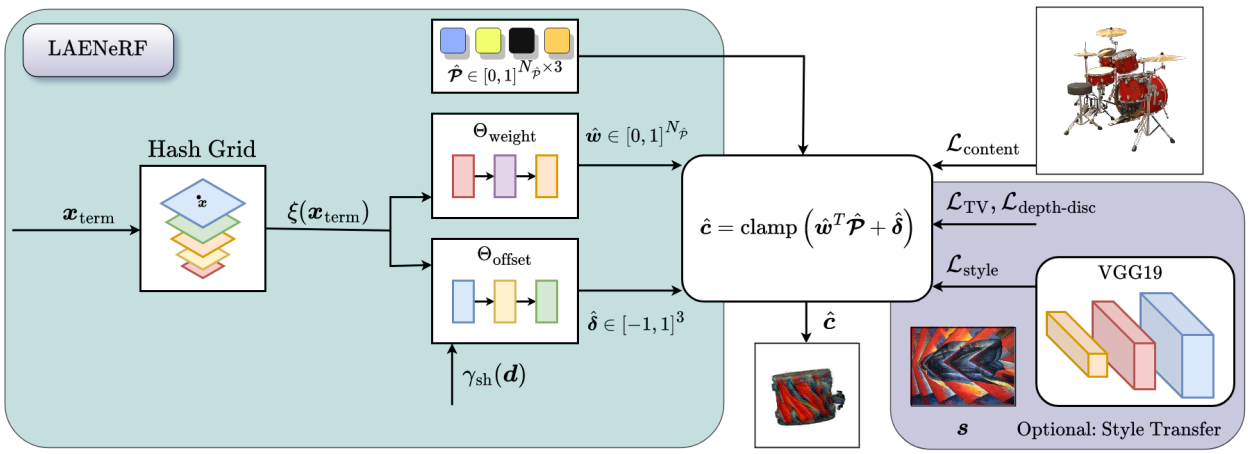

Figure 2. Overview of our method, LAENeRF. Given the estimated ray terminations $\boldsymbol{x}_{\text{term}}$ for a region specified by $\boldsymbol{\mathcal{E}}$, we learn a mapping from these inputs to barycentric weights $\hat{\boldsymbol{w}}$ and offsets $\hat{\boldsymbol{\delta}}$. We compose the per-ray color $\hat{\boldsymbol{c}}$ using these intermediate outputs and a learnable color palette $\hat{\boldsymbol{\mathcal{P}}}$ and supervise with a content loss $\mathcal{L}_{\text{content}}$ and optional style losses $\mathcal{L}_{\text{style}}, \mathcal{L}_{\text{TV}}, \mathcal{L}_{\text{depth-disc}}$ to obtain a unified approach which supports recoloring and stylization.

facilitate detailed, geometry-aware stylization, we introduce two novel losses, which condition LAENeRF on the estimated geometry of Θ_{NeRF}. First, we encourage LAENeRF to limit noise in regions where there are no depth discontinuities leveraging a depth guidance image:

$$(\nabla\zeta)_{i,j} = \left(|\zeta_{i,j+1} - \zeta_{i,j}|,\ |\zeta_{i+1,j} - \zeta_{i,j}|\right). \tag{9}$$

Then, we use this guidance image to restrict a total variation loss to regions without depth discontinuities. To this end, we introduce our novel, depth-guided total variation loss as

$$\mathcal{L}_{\text{TV}} = \lambda_{\text{TV}} \|\nabla\hat{\boldsymbol{c}} \cdot (1 - \nabla\zeta)\|_2^2, \tag{10}$$

where $\nabla\hat{\boldsymbol{c}}$ denotes the image gradients of $\hat{\boldsymbol{c}}$ in x/y-direction, i.e. $(\nabla\hat{\boldsymbol{c}})_{i,j} = \|\hat{\boldsymbol{c}}_{i+1,j} - \hat{\boldsymbol{c}}_{i,j}\|_2^2 + \|\hat{\boldsymbol{c}}_{i,j+1} - \hat{\boldsymbol{c}}_{i,j}\|_2^2$. This loss term remedies noisy prediction for $\boldsymbol{x}_{\text{term}}$, but does not preserve fine, geometric structures to a sufficient extent. Hence, we introduce another loss, which is minimized when image gradients are placed in regions of depth discontinuities:

$$\mathcal{L}_{\text{depth-disc}} = -\lambda_{\text{depth-disc}} \|\nabla\hat{\boldsymbol{c}} \cdot \nabla\zeta\|_2^2. \tag{11}$$

Palette Regularization. As we learn a palette-based decomposition of output colors given in Eq. (6), we require carefully designed regularization to ensure an intuitive color decomposition. We introduce a weight loss to encourage sparse per-pixel predictions:

$$\mathcal{L}_{\text{weight}} = \lambda_{\text{weight}} \left(1 - \|\hat{\boldsymbol{w}}\|_\infty\right). \tag{12}$$

To prevent extreme solutions with a high-frequency offset function dominating the color prediction [1], we regularize the offsets with

$$\mathcal{L}_{\text{offset}} = \lambda_{\text{offset}} \|\hat{\boldsymbol{\delta}}\|_2^2. \tag{13}$$

Finally, we regularize $\hat{\boldsymbol{\mathcal{P}}}$ to guarantee valid colors in RGB-space, i.e. $\hat{\boldsymbol{\mathcal{P}}}_{i,j} \in [0,1]$, using

$$\mathcal{L}_{\text{palette}} = \left\| \lfloor \hat{\boldsymbol{\mathcal{P}}} \rfloor \cdot \hat{\boldsymbol{\mathcal{P}}} \right\|_2^2. \tag{14}$$

Distillation of the Appearance Edits. To edit the apperance of a pre-trained radiance field, we first extract a training dataset for LAENeRF. After optimization, we can compose a modified training dataset for fine-tuning Θ_{NeRF}. We perform blending of the target colors $\mathcal{C}(\boldsymbol{r})$ and LAENeRF's output $\hat{\boldsymbol{c}}$ using the per-ray accumulated alpha as the blending weights.

Modelling Smooth Transitions. As our model operates on a user-defined region of interest and utilizes blending to construct a new dataset, recoloring or stylization produces sharp discontinuities on the boundary of $\boldsymbol{\mathcal{E}}$. While this behaviour is desirable when cells adjacent to $\boldsymbol{\mathcal{E}}$ are not occupied, it may lead to undesirable results otherwise. As the growing queue $\mathcal{G}$ stores adjacency information, we can use it to model smooth color transitions for more visually pleasing results. First, we construct a new grid $\boldsymbol{\mathcal{G}}$ from our growing queue $\mathcal{G}$. Then, we raymarch using $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{O}}$ to obtain the estimated ray terminations for each ray intersecting $\boldsymbol{\mathcal{G}}$, resulting in another point cloud, which we denote as $\mathcal{Z}$. For each $\boldsymbol{x}_{\text{term}}$ obtained from raymarching through $\boldsymbol{\mathcal{E}}$, we compute the minimum distance to $\mathcal{Z}$:

$$d_{\text{min}} = \min_{\boldsymbol{y} \in \mathcal{Z}} \|\boldsymbol{x}_{\text{term}} - \boldsymbol{y}\|_2. \tag{15}$$

We construct per-ray transition weights $d_{\text{trans}} \in [0,1]$ with

$$d_{\text{trans}} = 1 - \frac{\min(d_{\text{min}}, \tau_{\text{dist}})}{\tau_{\text{dist}}}, \tag{16}$$

where τ_{dist} is a hyperparameter controlling the size of the transition, which we usually set to 1×10^{-2}, depending on the scale of the scene. As can be seen in Fig. 3, we interpolate the original color palette $\hat{\mathcal{P}}$ and the modified color palette $\hat{\mathcal{P}}_{\text{mod}}$ based on d_{trans} to get a realistic color transition. For stylization, we additionally introduce a separate loss, which adds to the content loss when $d_{\text{trans}} \in (0, 1]$:

$$\mathcal{L}_{\text{smooth}} = \lambda_{\text{smooth}} \|(\hat{\boldsymbol{c}} - \boldsymbol{c})^2 \cdot d_{\text{trans}}\|_1. \tag{17}$$

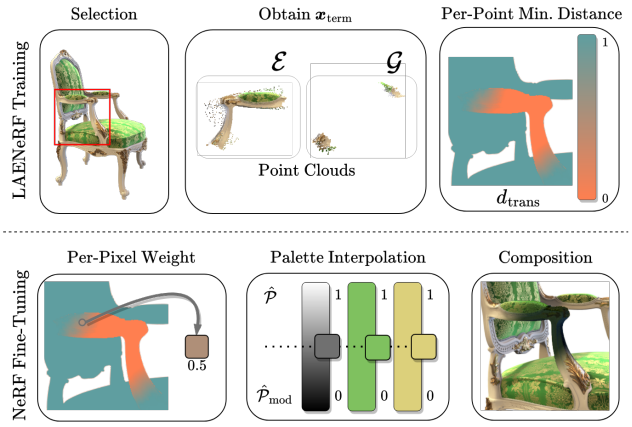

Figure 3. Illustration of our proposed distance-based palette interpolation scheme: We calculate distance weights d_{trans} based on the per-point distance from the edit grid $\boldsymbol{\mathcal{E}}$ to the growing grid $\boldsymbol{\mathcal{G}}$. When constructing the modified training dataset, we interpolate between learned palette $\hat{\mathcal{P}}$ and modified palette $\hat{\mathcal{P}}_{\text{mod}}$ using d_{trans}.

Implementation Details. We build our approach on torch-ngp [40], a PyTorch variant of iNGP [29]. For our edit grid $\boldsymbol{\mathcal{E}}$, we use a resolution of 128^3, the same as for the occupancy grid. We train LAENeRF for 1×10^5 iterations with previews available after $\sim$ 20s, and then distill to Θ_{NeRF} for 7×10^4 iterations, which takes no longer than 5 minutes in total on an NVIDIA RTX 4090. For LAENeRF, we use a learning rate of 1×10^{-3} for all components, except for $\hat{\mathcal{P}}$, where we use a learning rate of 1×10^{-2}. We initialize $N_{\hat{\mathcal{P}}} = 8$, a number which is usually too high for any scenario. However, we note that when we want to obtain a smooth transition utilizing $\mathcal{L}_{\text{smooth}}$, the increased number of base colors is vital. 1.5×10^3 iterations before training is finished, we remove color palettes which do not contribute significantly. For our style loss, we use a VGG19 backbone [36] and utilize features from `conv5`, `conv6`, `conv7` for computing $\mathcal{L}_{\text{style}}$. When performing stylization, we use $\lambda_{\text{style}} = 1.3 \times 10^2, \lambda_{\text{TV}} = 1 \times 10^{-4}, \lambda_{\text{depth-disc}} = 5 \times 10^{-4}, \lambda_{\text{weight}} = 1 \times 10^{-7}, \lambda_{\text{offset}} = 5 \times 10^{-5}, \lambda_{\text{smooth}} = 1 \times 10^{-3}$. If we want to perform photorealistic recoloring, we set $\lambda_{\text{style}} = \lambda_{\text{TV}} = \lambda_{\text{depth-disc}} = 0$.

5. Experiments

We use LAENeRF to perform local recoloring and local, recolorable stylization.

Datasets. We use three well-established datasets for novel view synthesis in our evaluation. **NeRF-Synthetic** [28] is a dataset consisting of synthetic objects with complex geometry and non-Lambertian materials. This dataset contains 360° captures in a bounded domain with a transparent background. **LLFF** [27] is a dataset consisting of forward-facing captures of real-world scenes in high resolution. The **mip-NeRF 360** dataset [3] contains 360° captures of unbounded indoor and outdoor scenes. This provides a challenging scenario for local appearance editing methods due to the large number of distinct objects in the scene and the large 3D space. For LLFF and mip-NeRF 360, we follow related work [20, 23] and downsample the images by a factor of $4\times$.

5.1. Photorealistic Appearance Editing

For the recoloring task, we compare our method with PaletteNeRF [20] and ICE-NeRF [23]. PaletteNeRF predicts features distilled from an LSeg segmentation model [19, 24] to enable local editing. In contrast, ICE-NeRF uses user-guided annotations to recolor a selected region given a target color.

Quantitative Evaluation. For the quantitative evaluation, we follow ICE-NeRF [23] and measure the Mean Squared Error (MSE) in the background of the selected region before and after recoloring. In Tab. 1, we present this metric for three scenes of the LLFF dataset [27]. The foreground region is described by a mask, which was provided to us by the authors of ICE-NeRF. We perform 7 different recolorings per scene and compare against PaletteNeRF with and without semantic guidance, whereas we also include the numbers from ICE-NeRF[1] to facilitate cross-method comparisons. LAENeRF outperforms previous methods for this metric, reducing error rates by 59% compared to PaletteNeRF with semantic guidance. We use Segment Anything [18] to obtain masks for a subset of test set views and compare against PaletteNeRF [20]. We measure MSE compared to the ground truth test images and use the same hyperparameters as PaletteNeRF, which also builds on torch-ngp [40]. As can be seen, LAENeRF after recoloring outperforms non-recolored PaletteNeRF, which we attribute to PaletteNeRF's concurrent scene reconstruction and palette-based decomposition. Due to the fairly accurate geometry for the 360° captures, our approach introduces very few background artefacts during recoloring.

[1] ICE-NeRFs' implementation is not publicly available at the time of submission.

Table 1. MSE (↓) in the background with respect to the unmodified images for our method, ICE-NeRF [23] and PaletteNeRF (PNF) [20] for the LLFF dataset [27]. Note that the recolorings from [23] are different to ours.

Scene	Results from [23]		Our Recolorings		
	PNF	ICE-NeRF	PNF	PNF (Semantic)	LAENeRF
Horns	0.0818	0.0213	0.0195	0.0028	**0.0010**
Fortress	0.0013	0.0010	0.0011	**0.0002**	**0.0002**
Flower	**0.0003**	**0.0003**	0.0076	0.0022	0.0007
Average	0.0277	0.0075	0.0094	0.0017	**0.0007**

Table 2. MSE (↓) in the background with respect to the test images for our method and PaletteNeRF [20] for the mip-NeRF 360 dataset [3]. LAENeRF exhibits lower error rates compared to PaletteNeRF, even when comparing trained with recolored.

Scene	PaletteNeRF			LAENeRF	
	Trained	Recolor	Recolor (Semantic)	Trained	Recolor
Bonsai	0.0015	0.0036	0.0016	0.0011	**0.0011**
Kitchen	0.0024	0.0125	0.0027	0.0021	**0.0022**
Room	0.0016	0.0216	0.0056	0.0015	**0.0015**
Average	0.0018	0.0124	0.0033	0.0016	**0.0016**

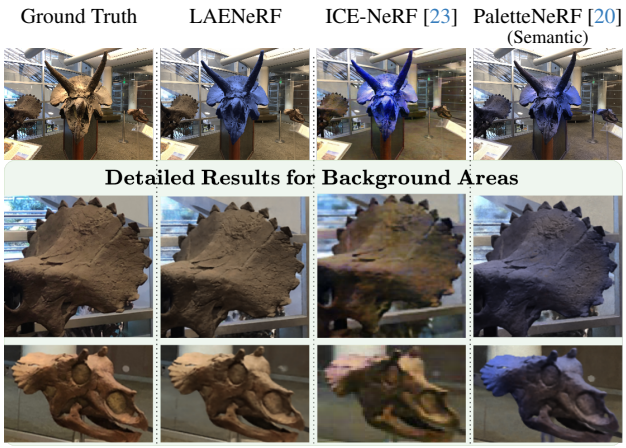

Figure 4. Qualitative comparison of our method with related work on the *Horns* scene of the LLFF dataset [27]. LAENeRF introduces far fewer artefacts compared to previous methods.

Qualitative Evaluation. In Fig. 4, we compare our method to ICE-NeRF [23] and PaletteNeRF [20]. As can be seen, our method introduces the fewest artefacts due to recoloring. To facilitate cross-method comparisons, we choose the same example as ICE-NeRF and include the results from their publication.

For the mip-NeRF 360 dataset [3], we compare against PaletteNeRF in Fig. 5. PaletteNeRF either introduces significant artefacts (see *Room*) or is unable to capture small regions with their semantic features (see *Bonsai*). As mentioned in their publication, ICE-NeRF struggles with local edits for this dataset, frequently introducing artefacts in undesired regions.

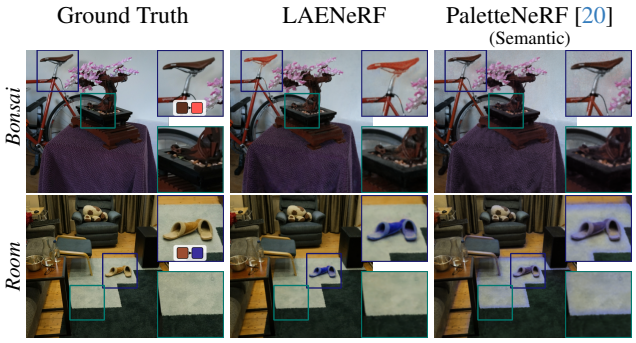

Figure 5. Qualitative comparison of our method to PaletteNeRF on the mip-NeRF 360 dataset [3] for small-scale edits. The top-row detailed view shows the selected region for recoloring, whereas the bottom-row view shows a background region. Our method introduces fewer errors in the background whilst recoloring the selected object faithfully.

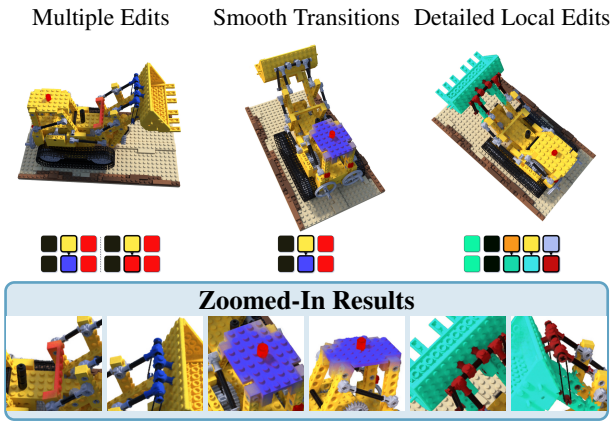

Figure 6. Demonstration of our editing capabilities: LAENeRF can perform arbitrary edits on any local region with smooth transitions.

In Fig. 6, we show some recoloring results only possible with our method on the synthetic *Lego* scene. PaletteNeRF's semantic features do not permit any of the shown edits. While ICE-NeRF can perform multiple color edits, their approach recolors based on a single target color and thus fails for edits where multiple palette changes are required. LAENeRF is the only method which can model smooth transitions between original scene content and the recolored region.

5.2. Non-Photorealistic Appearance Editing

For style transfer, we compare our method to Ref-NPR [54], which stylizes a scene based on one or a few reference images. As Ref-NPR is not specifically designed for local stylization, we create locally stylized reference images by selecting three training dataset images, applying AdaIN [14] for stylization using a style image s, and using LAENeRF's blending weights to generate three reference images. Additional details are available in the supplementary material.

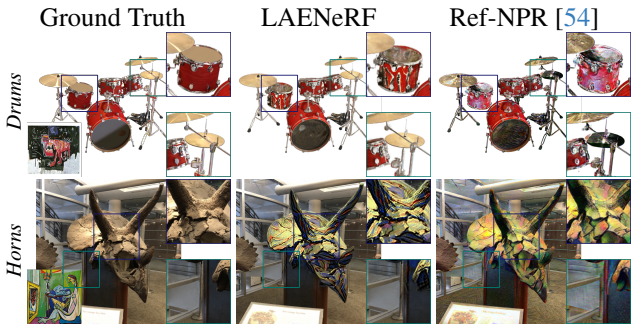

Figure 7. Qualitative comparison of our method to Ref-NPR. The top-row detailed view shows the selected region for stylization, whereas the bottom-row view shows a background region. Our method produces a more detailed stylization while minimizing background artefacts.

Quantitative Evaluation. For our quantitative evaluation, we measure MSE in the background with respect to the ground truth test set images. For NeRF-Synthetic [28], we generate masks for the region to stylize using our method and use segmentation masks from ICE-NeRF [23] for LLFF [28] scenes. We report per-dataset results in Tab. 3. In contrast to Ref-NPR, our approach demonstrates considerably reduced error rates, especially in synthetic scenes characterized by numerous occlusions.

For forward-facing scenes, Ref-NPR benefits from less variation between camera poses but still generates about 3× more errors than ours.

Table 3. MSE (↓) in the background with respect to the ground truth test images for our method and Ref-NPR [54]. LAENeRF significantly outperforms Ref-NPR for synthetic and forward-facing scenes.

Dataset	Ref-NPR	LAENeRF
NeRF-Synthetic [28]	0.0466	**0.0071**
LLFF [27]	0.0073	**0.0025**
Average	0.0270	**0.0048**

Qualitative Evaluation. In Fig. 7, we show results for our method and Ref-NPR for synthetic and real-world scenes. Our approach introduces fewer background artefacts compared to Ref-NPR, while stylizing the selected region with more detail. In Fig. 9, we show additional results on local, recolorable stylization for all scene types, including unbounded, real-world scenes [3]. As can be seen, our approach is compatible with diverse datasets, stylizes the selected region faithfully and produces intuitive color palettes for interactive recoloring.

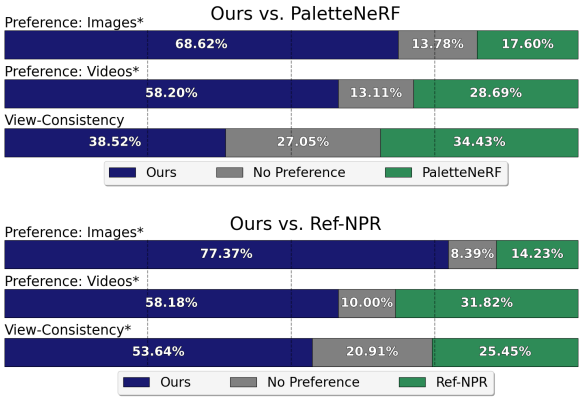

Figure 8. User study results. Participants prefer our method to related work for image and video outputs. Our method is also rated higher for view-consistency than Ref-NPR [54]. * indicates a statistical significance according to Wilcoxon signed rank tests.

5.3. User Study

To further evaluate our approach, we conducted a user study comparing our approach to PaletteNeRF [20] and Ref-NPR [54]. We showed the participants pairs of recolored/stylized images with recoloring/style and target region inset and asked which result they preferred (a, b, or no-preference). Additionally, we presented pairs of recolored/stylized videos of the same scenes and inquired about preference for visual quality and view-consistency (without reference). We collected 847 responses from 31 participants as summarized in Fig. 8. Participants prefer our approach for both images and videos and rated our approach as more view-consistent than Ref-NPR. We refer to the supplementary material for details.

5.4. Time Comparisons

We use the *Flower* scene of the LLFF dataset [27], for an exemplary time comparison: With a pre-trained radiance field, PaletteNeRF [20] requires 13.5 min for recoloring a selected object, whereas our method accomplishes the same task in 3 min. When provided with stylized reference views, Ref-NPR [54] achieves scene stylization in 2.5 min, whereas our approach takes 2 min. More importantly, we always provide previews after ∼ 20s.

Figure 9. Results for local stylization for various scene types. LAENeRF can faithfully transfer the style of an arbitrary style image to a selected region whilst minimizing errors in the background. Due to our decomposition into base colors, stylized regions remain editable.

6. Limitations

Although LAENeRF is a flexible method for local appearance edits of NeRF, some challenges remain. Similar to [32], our ability to modify stylized regions is restricted, specifically to adjusting palette bases post-training. We leverage the pre-trained NeRF to perform geometry-aware appearance modifications. As noted in other works [3, 30, 33], radiance fields often trade geometric fidelity for visual quality by modelling non-Lambertian effects with additional samples behind the surface, posing a challenge to our point-based optimization scheme. Particularly for real-world scenes, this may lead to reduced quality in the edited region. Another disadvantage of LAENeRF lies in the separation of optimization and distillation. This design choice allows for interactive recoloring of stylized content as an intermediate step but incurs additional time for generating a modified training dataset and NeRF fine-tuning.

7. Conclusion

We present LAENeRF, a unified framework for photorealistic and non-photorealistic appearance editing of NeRF. By elegantly combining a palette-based decomposition with perceptual losses, we enable interactive recoloring of stylized regions. We demonstrate state-of-the-art local appearance editing results, benefitting from our geometry-aware stylization in 3D. LAENeRF outperforms existing works quantitatively and qualitatively for local recoloring and local stylization. By open-sourcing our approach we will bring NeRF-editing to a large audience.

References

[1] Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM TOG*, 36(2):61c:1–61c:19, 2017. 4

[2] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. SINE: Semantic-driven Image-based NeRF Editing with Prior-guided Editing Field. In *CVPR*, 2023. 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 *CVPR*, 2022. 5, 6, 7, 8, 13, 15, 16

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*, 2023. 1

[5] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. In *CVPR*, 2021. 2

[6] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated Neural Radiance Fields in the Wild. In *CVPR*, 2022. 1

[7] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified Implicit Neural Stylization. In *ECCV*, 2022. 2

[8] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*, 2022. 3

[9] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*, 2016. 2, 3

[10] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling Perceptual Factors in Neural Style Transfer. In *CVPR*, 2017. 2

[11] Bingchen Gong, Yuehao Wang, Xiaoguang Han, and Qi Dou. RecolorNeRF: Layer Decomposed Radiance Fields for Efficient Color Editing of 3D Scenes. In *ACM MM*, 2023. 2

[12] Ori Gordon, Omri Avrahami, and Dani Lischinski. Blended-NeRF: Zero-Shot Object Generation and Blending in Existing Neural Radiance Fields. In *ICCV AI3DCC*, 2023. 2

[13] Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. Learning to Stylize Novel Views. In *CVPR*, 2021. 2

[14] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *CVPR*, 2017. 2, 7, 13, 14

[15] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. StylizedNeRF: Consistent 3D Scene Stylization as Stylized NeRF via 2D-3D Mutual Learning. In *CVPR*, 2022. 2

[16] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. NeRFshop: Interactive Editing of Neural Radiance Fields. *ACM CGIT*, 6(1):1:1–1:21, 2023. 2, 3, 13

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *ECCV*, 2016. 2, 3

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment Anything. In *ICCV*, 2023. 5, 15

[19] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. In *NeurIPS*, 2022. 1, 2, 5

[20] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. PaletteNeRF: Palette-based Appearance Editing of Neural Radiance Fields. In *CVPR*, 2023. 2, 5, 6, 7, 11, 12, 13, 14, 15, 16

[21] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. AdaNeRF: Adaptive Sampling for Real-time Rendering of Neural Radiance Fields. In *ECCV*, 2022. 1

[22] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning Blind Video Temporal Consistency. In *ECCV*, 2018. 12

[23] Jae-Hyeok Lee and Dae-Shik Kim. ICE-NeRF: Interactive Color Editing of NeRFs via Decomposition-Aware Weight Optimization. In *ICCV*, 2023. 1, 2, 5, 6, 7, 14

[24] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven Semantic Segmentation. In *ICLR*, 2022. 5

[25] Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric P Xing. StyleRF: Zero-shot 3D Style Transfer of Neural Radiance Fields. In *CVPR*, 2023. 2, 12

[26] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 1

[27] 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 TOG*, 38(4):29:1–29:14, 2019. 5, 6, 7, 13, 14, 15, 16

[28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 1, 3, 5, 7, 13, 15

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4):102:1–102:15, 2022. 2, 3, 5, 11

[30] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Comput. Graph. Forum*, 40(4):45–59, 2021. 8

[31] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. SNeRF: Stylized Neural Implicit Representations for 3D Scenes. *ACM TOG*, 41(4):142:1–142:11, 2022. 2, 12

[32] Hong-Wing Pang, Binh-Son Hua, and Sai-Kit Yeung. Locally Stylized Neural Radiance Fields. In *ICCV*, 2023. 2, 8

[33] Julien Philip and Valentin Deschaintre. Floaters No More: Radiance Field Gradient Scaling for Improved Near-Camera Training. In *EGSR*, 2023. 8

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2022. 1

[35] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic Style Transfer for Videos. In *GCPR*, 2016. 12

[36] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2014. 3, 5

[37] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing Images into Layers via RGB-Space Geometry. *ACM TOG*, 36(1):7:1–7:14, 2016. 2

[38] Jianchao Tan, Jose Echevarria, and Yotam Gingold. Efficient Palette-Based Decomposition and Recoloring of Images via RGBXY-Space Geometry. *ACM TOG*, 37(6):262:1–262:10, 2018. 2

[39] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *CVPR*, 2022. 1

[40] Jinxiang Tang. Torch-ngp: a PyTorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 5, 11, 13

[41] Jiaxiang Tang, Hang Zhou, Xiaokang Chen, Tianshu Hu, Errui Ding, Jingdong Wang, and Gang Zeng. Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement. In *ICCV*, 2022. 2

[42] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*, 2020. 12

[43] 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

[44] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *CVPR*, 2022. 2, 3

[45] Dor Verbin, Ben Mildenhall, Peter Hedman, Jonathan T. Barron, Todd Zickler, and Pratul P. Srinivasan. Eclipse: Disambiguating Illumination and Materials using Unintended Shadows. *arXiv CoRR abs:2305.16321*, 2023. 2

[46] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. NeRF-Art: Text-Driven Neural Radiance Fields Stylization. In *IEEE TVCG*, 2023. 2

[47] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. InpaintNeRF360: Text-Guided 3D Inpainting on Unbounded Neural Radiance Fields. *arXiv CoRR abs:2305.15094*, 2023. 2

[48] R. F. Woolson. *Wilcoxon Signed-Rank Test*, pages 1–3. John Wiley & Sons, Ltd, 2008. 13

[49] Qiling Wu, Jianchao Tan, and Kun Xu. PaletteNeRF: Palette-based Color Editing for NeRFs. *arXiv CoRR abs:2212.12871*, 2022. 2

[50] 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. In *ICCV*, 2023. 2

[51] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. ARF: Artistic Radiance Fields. In *ECCV*, 2022. 2

[52] 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. 12

[53] 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 TOG*, 40(6):237:1–237:18, 2021. 2

[54] Yuechen Zhang, Zexin He, Jinbo Xing, Xufeng Yao, and Jiaya Jia. Ref-NPR: Reference-Based Non-Photorealistic Radiance Fields for Controllable Scene Stylization. In *CVPR*, 2023. 2, 7, 12, 13, 14, 15

[55] Zicheng Zhang, Yinglu Liu, Congying Han, Yingwei Pan, Tiande Guo, and Ting Yao. Transforming Radiance Field with Lipschitz Network for Photorealistic 3D Scene Stylization. In *CVPR*, 2023. 2

[56] Chengwei Zheng, Wenbin Lin, and Feng Xu. EditableNeRF: Editing Topologically Varying Neural Radiance Fields by Key Points. In *CVPR*, 2023. 1

[57] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant Volumetric Head Avatars. In *CVPR*, 2023. 1

LAENeRF: Local Appearance Editing for Neural Radiance Fields

Supplementary Material

A. Additional Implementation Details

In this section, we present more information on the network architecture of LAENeRF.

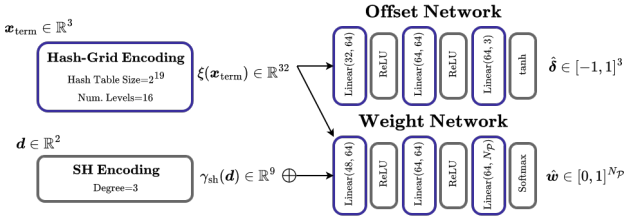

Figure 10. **Network architecture** for LAENeRF. Blue components indicate trainable parameters and the input for the weight network is padded with ones.

A.1. Network Architecture

In Fig. 10, we show the network architecture for LAENeRF. As discussed in our main material, our module is NeRF-like: The sizes of the hash grid and linear layers as well as the spherical harmonics encoding are used similarly in iNGP [29]. However, instead of two subsequent MLPs predicting density and color, we split our model and evaluate the weight and the offset network concurrently.

A.2. Removal of Base Color Palettes

As indicated in the main material, we start with $N_{\hat{\mathcal{P}}} = 8$ base colors, which are initialized according to a uniform distribution, i.e. $\hat{\mathcal{P}} \sim \mathcal{U}[0, 1)$. Because 8 color palettes are usually only required when we want to obtain smooth transitions during stylization, we remove color palettes which do not contribute significantly before the final 1500 iterations. To do this, we sample 10 poses from the training dataset and evaluate our weight network. We derive a per-palette mean contribution, which is always between 0 and 1. Finally, we choose a threshold of 0.025 and remove all color palettes which contribute less, which we reflect by updating our UI.

A.3. Geometry-Aware Stylization

To perform stylization which respects the learned geometry of our pre-trained NeRF, we perform depth estimation. As we set the depth to 0 for all rays which did not intersect $\boldsymbol{\mathcal{E}}$, we would get incorrect results when computing our depth guidance image. Additionally, direct neighbors of rays which did not intersect the edit grid often do not accumulate full alpha inside $\boldsymbol{\mathcal{E}}$, leading to incorrect predictions for ζ. To remedy the aforementioned issues, we multiply $\nabla\zeta$ with the accumulated alpha inside the edit grid $\alpha^{(\boldsymbol{\mathcal{E}})}$, for both pixels involved in the computation and their direct neighbors:

$$(\nabla\zeta)_{i,j} = (\nabla\zeta)_{i,j} \cdot \left(\prod_{y=j-1}^{j+2} \alpha_{i,y}^{(\boldsymbol{\mathcal{E}})}, \prod_{z=i-1}^{i+2} \alpha_{z,j}^{(\boldsymbol{\mathcal{E}})} \right). \quad (18)$$

A.4. Modifications to our Framework

We include the following modifications to torch-ngp [40] to improve reconstruction for real-world scenes, following PaletteNeRF [20]. By default, the background color is assumed to be white for real-world scenes. We use a random background color during training, which leads to better depth estimation and a sparser occupancy grid. In addition, we mark grid cells between each camera and its' corresponding near plane as non-trainable.

B. Ablation Studies

In this section, we provide additional analysis for the individual components of LAENeRF. In particular, we focus on the proposed losses for stylization and regularization of the color palette.

B.1. Color Palette Regularization

To measure the fidelity of the learned color palette $\hat{\mathcal{P}}$ quantitatively, we adopt the metrics from PaletteNeRF [20] and measure sparsity with

$$\mathcal{L}_{\text{sp}} = \frac{\sum_{i=1}^{N_{\hat{\mathcal{P}}}} \hat{\boldsymbol{w}}_i}{\sum_{i=1}^{N_{\hat{\mathcal{P}}}} \hat{\boldsymbol{w}}_i^2} - 1. \quad (19)$$

In addition, we measure the total variation of the per-palette weight images $\hat{\boldsymbol{w}}_i$. We present the metrics in Tab. 4, where we recolored/stylized the shovel of the bulldozer in the *Lego* scene. Not using $\mathcal{L}_{\text{offset}}$ causes extreme solutions, evidenced by $\mathcal{L}_{\text{sp}}$ and the non-intuitive recoloring in Fig. 11. Not using $\mathcal{L}_{\text{weight}}$ extracts intuitive palettes but uses more base colors and results in high values for $\mathcal{L}_{\text{sp}}$. LAENeRF achieves the best sparsity and requires the fewest base colors.

B.2. Geometry-Aware Stylization Losses

Naïvely applying 2D style transfer losses often results in removal of smaller structure, *e.g.* the black rubber bands in the *Lego* scene disappear to obtain a more coherent stylization. We demonstrate the effectiveness of our proposed countermeasures in Fig. 12. As can be seen, LAENeRF retains

Table 4. **Quantitative Evaluation** of our proposed color palette regularization. Our full model achieves the best metrics for sparsity and uses the fewest base colors.

Method	Stylization			Recoloring		
	$N_{\hat{\mathcal{P}}}$	$\mathcal{L}_{\text{sp}} \downarrow$	TV $\downarrow$	$N_{\hat{\mathcal{P}}}$	$\mathcal{L}_{\text{sp}} \downarrow$	TV $\downarrow$
w/o $\mathcal{L}_{\text{offset}}$	8	6.046	**0.005**	8	6.304	**0.005**
w/o $\mathcal{L}_{\text{weight}}$	8	3.033	0.026	8	2.375	0.042
LAENeRF	**6**	**2.129**	0.029	**5**	**0.859**	0.107

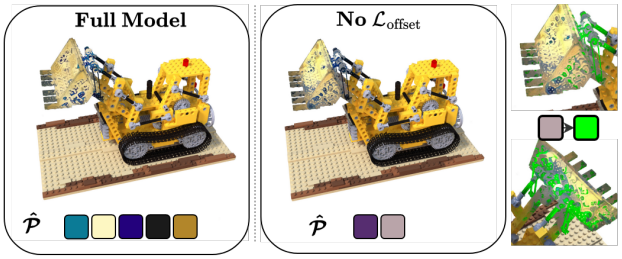

Figure 11. **Ablation study** on the effect of $\mathcal{L}_{\text{offset}}$. Without this loss term, our model suffers from incorrect palette reconstruction and highly non-intuitive recoloring.

small structures well during stylization compared to only using $\mathcal{L}_{\text{TV}}$ and the naïve approach without any geometry-conditioned losses. When stylizing, we additionally train LAENeRF without $\mathcal{L}_{\text{style}}, \mathcal{L}_{\text{offset}}, \mathcal{L}_{\text{TV}}$ for the first 1000 iterations, resulting in a well-initialized palette $\hat{\mathcal{P}}$.

B.3. View-Consistency

In addition to our experiments in the main paper, we evaluate a view-consistency metric from SNeRF [31], which is a generalization of Lai *et al*. [22] to the NeRF setting. Following SNeRF, we estimate the optical flow using RAFT [42] and use the occlusion detection method from Ruder *et al*. [35] to derive an occlusion mask O. Importantly, we compute the optical flow on a video sequence rendered from a pre-trained NeRF. Then, we compute the MSE between the warped view $\bar{\boldsymbol{V}}_{i+\delta}$ and the rendered view $\boldsymbol{V}_{i+\delta}$ using

$$E_{\text{warp}}(\boldsymbol{V}_{i+\delta}, \bar{\boldsymbol{V}}_{i+\delta}) = \frac{1}{|O|} \left\| \bar{\boldsymbol{V}}_{i+\delta} - \boldsymbol{V}_{i+\delta} \right\|_2^2. \tag{20}$$

In addition, we also measure LPIPS [52] between $\bar{\boldsymbol{V}}_{i+\delta}$ and $\boldsymbol{V}_{i+\delta}$, following StyleRF [25]. We show our results in Tab. 5, where we analyzed 6 video sequences for stylization and recoloring. We evaluate short-range consistency ($\delta = 1$) and long-range consistency ($\delta = 7$). LAENeRF outperforms Ref-NPR [54] for all metrics. PaletteNeRF [20] with semantic features achieves slightly better results for MSE, but performs worse for LPIPS.

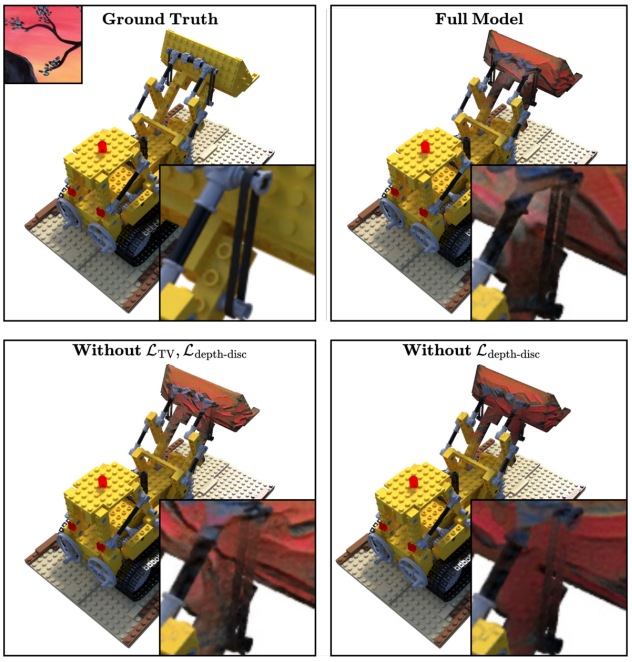

Figure 12. **Ablation study** on the effectiveness of our proposed losses for 3D-aware stylization. Our full model retains the most detail during stylization.

These quantitative results align with the results for our user study.

Table 5. **View-Consistency comparison** of our method, Ref-NPR [54] and PaletteNeRF [20] with semantic guidance. We measure short-range consistency ($\delta = 1$) and long-range consistency ($\delta = 7$).

	Stylization Consistency			
	Short-Range		Long-Range	
	MSE (↓)	LPIPS (↓)	MSE (↓)	LPIPS (↓)
LAENeRF	**0.0252**	**0.0650**	**0.1932**	**0.2253**
Ref-NPR	0.0264	0.0722	0.1973	0.2482
	Recoloring Consistency			
	Short-Range		Long-Range	
	MSE (↓)	LPIPS (↓)	MSE (↓)	LPIPS (↓)
LAENeRF	0.0167	**0.0587**	0.0934	**0.2063**
PaletteNeRF (semantic)	**0.0164**	0.0634	**0.0925**	0.2080

C. Additional User Study Details

We conduct our user study for a comparison with state-of-the-art methods for local recoloring and local style transfer in scenes represented by NeRFs. We conduct the study on

an iPad 9th Gen with a 10.2" display. All images and videos used for our user study are included in the supplementary material.

Images: Local Recoloring. For local recoloring, we select scenes from the LLFF dataset [27] (*Flower, Horns, Fortress, Orchids, Trex*) and the mip-NeRF 360 dataset [3] (*Kitchen, Bonsai, Room*) and compare against PaletteNeRF [20] with semantic guidance. For the per-image comparisons, we prepare 11 different recolorings, where we tried to align the results as much as possible for a fair comparison. For each pair of images (one from our method, one from PaletteNeRF), the user is shown the reference test set image, the region we want to recolor and a color change. We randomize the order in which we present the different conditions. Users were instructed to choose the image they prefer (a, b, or no-preference) based on background artefacts, image quality and personal preference.

Images: Local Stylization. For local stylization, we select scenes from the LLFF dataset [27] (*Flower*, *Horns*, *Trex*) and the NeRF-Synthetic dataset [28] (*Lego, Hotdog, Chair, Drums*) and compare against Ref-NPR [54]. For the per-image comparison, we prepare 9 different stylizations, where we tried to align the results as much as possible for a fair comparison. Instead of showing the user a color change, we now show the user the style image $\boldsymbol{s}$. The images for Ref-NPR were generated leveraging AdaIN [14], as shown in Fig. 13. The testing modality is identical to local recoloring, however, users were also instructed to consider the transfer of style from the given style image $\boldsymbol{s}$ to the selected region.

Videos. In addition to images, users were also shown videos of recolorings and stylizations. For this modality, we did not provide a specified region, target color or style images. Users were instead instructed to select their preferred video based on (1) visual appearance and (2) view-consistency.

Participant Details. Of the 31 participants of our user study, 58% had at least some prior experience with NeRFs. In addition, 29% used a visual aid during the user study.

Statistical Evaluation. We compute the Wilcoxon signed rank test [48] to determine whether there is statistical significance in the preference for one or the other method. All preference scores indicate a significant preference for LAENeRF with recoloring ($Z = 8850.0;\ p < 0.0001$), stylization ($Z = 5000.0;\ p < 0.0001$), video recoloring ($Z = 1872.5;\ p < 0.0005$), and video stylization ($Z = 1750.0;\ p < 0.005$). The view consistency scores indicate a preference for our method for stylization ($Z = 1232.0;\ p < 0.001$), but no significant difference for recoloring ($Z = 1890.0;\ p > 0.5$).

D. GUI

LAENeRF incorporates a GUI building on [16, 40] for real-time, interactive appearance editing of NeRFs. Here, we present several key components of our user interface in detail.

D.1. Region Selection and Growing

We provide an intuitive region selection process, based on NeRFShop [16]. First, the user scribbles on the image rendered from the current camera pose in our real-time viewer. For each selected ray, we perform raymarching and map the estimated ray termination to a the nearest voxel, which we set in $\boldsymbol{\mathcal{E}}$, representing our initial selection. Next, the initial selection can be extended with region growing, where we control growing by the number of voxels which we pop from our growing queue per-iteration and the total number of iterations. Our GUI shows the selection after each region growing step. We can optionally create a growing grid $\boldsymbol{\mathcal{G}}$ from our selection to model smooth transitions. Further, we incorporate binary operations with a second grid for more intuitive addition and deletion of selected voxels. The complete workflow can be seen in Fig. 14.

D.2. Style Image Selection

Our LAENeRF module requires style images $\boldsymbol{s} \in \mathbb{R}^{256\times256\times3}$ for stylization. To enhance usability of our approach and to support arbitrary style images, we provide an intuitive GUI for style image selection. As can be seen in Fig. 15, our GUI supports zooming and cropping arbitrary images to the required size.

D.3. More Palette Control

To provide users with more control over the stylized/recolored region, we enable a linear transformation of the learned weights $\hat{\boldsymbol{w}}$. To this end, we introduce palet weights $\boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}} \in \mathbb{R}^{N_{\hat{\mathcal{P}}}}$ and palette biases $\boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}} \in [-1,1]^{N_{\hat{\mathcal{P}}}}$, which transform the weights according to

$$\begin{aligned} \hat{\boldsymbol{w}} &= \min(\hat{\boldsymbol{w}} \cdot \boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}} + \boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}}, 0), \\ \hat{\boldsymbol{w}} &= \frac{\hat{\boldsymbol{w}}}{\mathbf{1}^T \hat{\boldsymbol{w}}}. \end{aligned} \tag{21}$$

We initialize $\boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}} = \mathbf{1},\ \boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}} = \mathbf{0}$ and let the user guide these parameters after LAENeRF is fully-trained, as can be seen in Fig. 16.

D.4. Preview Mode & GUI demonstration

To enable interactive recoloring and stylization, our approach supports real-time rendering during training of our

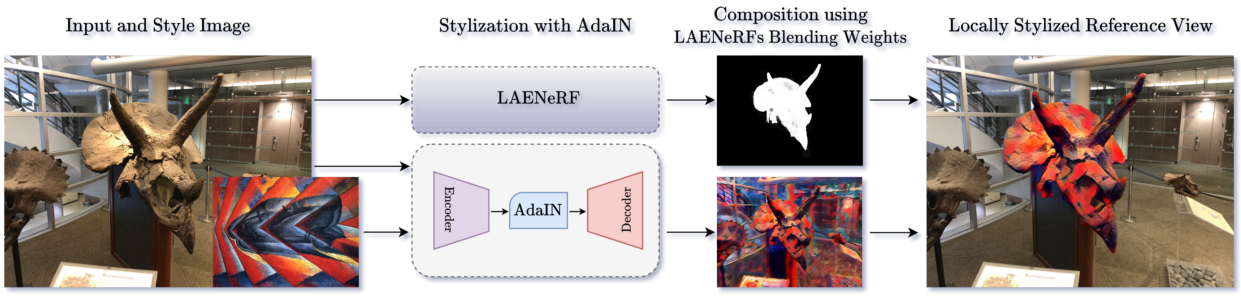

Figure 13. **Generation of locally stylized reference images** for Ref-NPR [54] using AdaIN [14] and LAENeRF's blending weights.

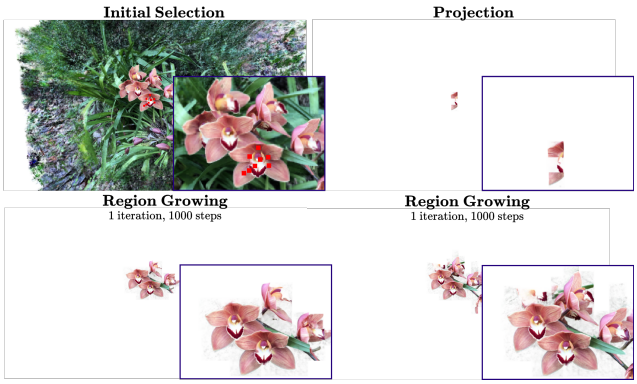

Figure 14. **Region selection** with our interactive GUI. The user first clicks on the screen (shown as red squares). For each selected ray, we compute the estimated ray termination and map it to the nearest cell in $\mathcal{E}$. Finally, this initial selection can be extended with region growing (2 iterations shown here).

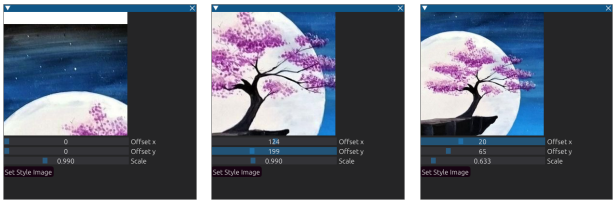

Figure 15. **GUI interface** for style image selection.

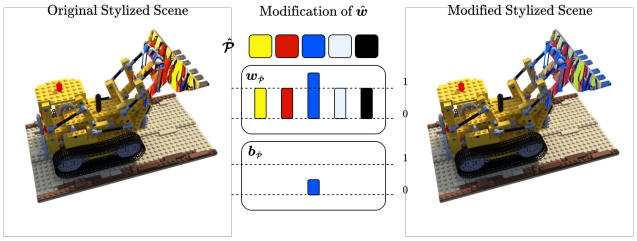

Figure 16. **Demonstration of user-guided modifications to $\hat{w}$.** We enable modification of stylized regions by changing the importance of individual base colors $\hat{\mathcal{P}}_i$ via modification of $\boldsymbol{w}_{\hat{\mathcal{P}}}, \boldsymbol{b}_{\hat{\mathcal{P}}}$.

LAENeRF module. After the training dataset has been extracted, which takes $\sim$ 15 seconds depending on the dataset, users can watch our neural module converge. We prepare a video demonstration of our application, which is contained in the supplementary material. This demonstration also includes more recoloring and stylization results. The video is available here: https://youtu.be/iXVr6OCD_CA.

D.5. Detailed Time Comparisons

We provide a more detailed analysis of the time comparison to PaletteNeRF [20] in Tab. 6. As can be seen, LAENeRF performs the recoloring task much faster, although the same NeRF backbone is used.

Table 6. **Detailed timing comparison** of our method and PaletteNeRF [20].

PaletteNeRF					
Semantic Features	Train NeRF	Extract Palette	Train PaletteNeRF		Total
90s	153s	10s	572s		815s
LAENeRF					
Train NeRF	Edit Dataset	Train LAENeRF	Distill Dataset	Fine-Tune NeRF	Total
153s	16s	133s	32s	129s	**463s**

E. Per Scene Results for Quantitative Evaluation

For a better reasoning behind the quantitative results in the main paper and to facilitate more research in the area of local appearance editing, we report per scene results for recoloring and stylization.

E.1. LLFF Recoloring

For the LLFF dataset [27], we show our per-recoloring results in Tab. 8. All masks were provided to us by the authors of ICE-NeRF [23]. For our color edits, we also in-

Figure 17. **Example of Color Edits** for the LLFF dataset [27] for our method and PaletteNeRF [20].

clude substantial recolorings, as can be seen in Fig. 17. PaletteNeRF [20] requires changing all color palettes for this specific example due to their decomposition, which introduces background artefacts, even when semantic features are used for guidance. Our approach geometrically segments the region in 3D, leading to good recoloring results without introducing significant artefacts in the background.

E.2. mip-NeRF 360 Recoloring

For the mip-NeRF 360 dataset [3], we report per-recoloring results in Tab. 9. We measure MSE in the background of the selected region, with masks extracted using Segment Anything [18] describing the selected object. We obtain 14 masks for test scenes of the *Bonsai* scene, 32 masks for the *Kitchen* scene and 18 masks for the *Room* scene. We show some examples of the extracted masks in Fig. 18.

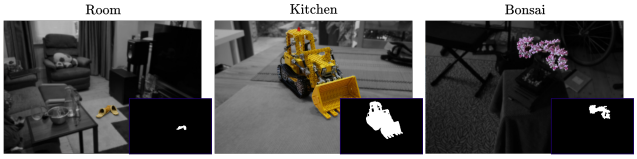

Figure 18. **Foreground masks** for the mip-NeRF 360 dataset [3]. We lowered brightness and saturation and additionally blurred the background using the masks.

E.3. Local Stylization

For local stylization, we report per scene results in Tab. 7. As can be seen, our method outperforms Ref-NPR [54] for every scene.

Table 7. MSE (↓) in the background per scene for local stylization for our method and Ref-NPR [54].

Method	NeRF-Synthetic [28]			
	Chair	*Drums*	*Lego*	Average
Ref-NPR [54]	0.0357	0.0808	0.0233	0.0466
LAENeRF	0.0037	0.0154	0.0021	**0.0071**
	LLFF [27]			
	Horns	*Flower*	*Trex*	Average
Ref-NPR [54]	0.0075	0.0079	0.0064	0.0073
LAENeRF	0.0024	0.0026	0.0025	**0.0025**

Table 8. MSE (↓) in the background per recoloring for the LLFF dataset [27] for our method and PaletteNeRF [20] with and without semantic guidance.

	Horns							
	cyan	dark red	green	grey	magenta	light red	yellow	Average
PaletteNeRF	0.0026	0.0437	0.0138	0.0347	0.0004	0.0237	0.0173	0.0195
PaletteNeRF (semantic)	0.0008	0.0084	0.0011	0.0072	0.0004	0.0016	0.0001	0.0028
LAENeRF	0.0010	0.0010	0.0010	0.0010	0.0008	0.0011	0.0007	**0.0010**
	Fortress							
	dark turquoise	magenta	dark orange	white	indigo	blueviolet	greenyellow	Average
PaletteNeRF	0.0007	0.0009	0.0013	0.0010	0.0009	0.0018	0.0011	0.0011
PaletteNeRF (semantic)	0.0001	0.0001	0.0001	0.0001	0.0004	0.0002	0.0002	**0.0002**
LAENeRF	0.0002	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002	**0.0002**
	Flower							
	dark purple	light red	green	yellow	green blue	orange	purple	Average
PaletteNeRF	0.0172	0.0005	0.0057	0.0099	0.0076	0.0100	0.0026	0.0076
PaletteNeRF (semantic)	0.0102	0.0004	0.0009	0.0010	0.0012	0.0004	0.0012	0.0022
LAENeRF	0.0006	0.0006	0.0007	0.0010	0.0006	0.0006	0.0006	**0.0007**

Table 9. MSE (↓) in the background per recoloring for the mip-NeRF 360 dataset [3] for our method and PaletteNeRF [20] with and without semantic guidance. LAENeRF outperforms PaletteNeRF for every recoloring, demonstrating effectiveness for diverse recolorings.

	Room, Sandles						
	blue	green	red	dark grey	purple	greenyellow	Average
PaletteNeRF	0.0384	0.0186	0.0046	0.0339	0.0291	0.0050	0.0216
PaletteNeRF (semantic)	0.0046	0.0027	0.0023	0.0136	0.0059	0.0047	0.0056
LAENeRF	0.0015	0.0015	0.0015	0.0015	0.0015	0.0015	**0.0015**
	Bonsai, Flower						
	blueviolet	light yellow	red	lime	purple	black	Average
PaletteNeRF	0.0025	0.0034	0.0040	0.0038	0.0028	0.0048	0.0036
PaletteNeRF (semantic)	0.0015	0.0015	0.0015	0.0016	0.0016	0.0016	0.0016
LAENeRF	0.0011	0.0011	0.0011	0.0011	0.0011	0.0012	**0.0011**
	Kitchen, Bulldozer						
	dark turquoise	magenta	dark orange	white	indigo	blueviolet	Average
PaletteNeRF	0.0044	0.0039	0.0025	0.0038	0.0034	0.0568	0.0125
PaletteNeRF (semantic)	0.0028	0.0028	0.0024	0.0027	0.0027	0.0027	0.0027
LAENeRF	0.0023	0.0022	0.0021	0.0021	0.0021	0.0021	**0.0022**