LAENeRF: Local Appearance Editing for Neural Radiance Fields Supplementary Material

Lukas Radl[1] Michael Steiner[1] Andreas Kurz[1] Markus Steinberger[1,2]
{lukas.radl, michael.steiner, andreas.kurz, steinberger}@icg.tugraz.at
[1]Graz University of Technology [2]Huawei Technologies, Austria

In this supplementary document, we provide more implementation details and quantitative results. Additionally, we describe our GUI in detail and provide additional details for our user study.

1. Additional Implementation Details

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

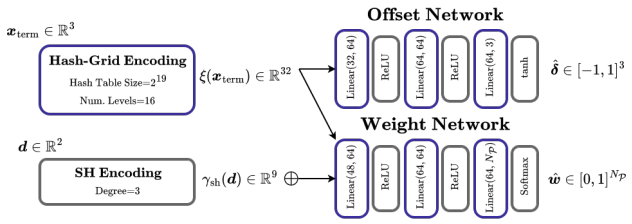

Figure 1. **Network architecture** for LAENeRF. Blue components indicate trainable parameters and the input for the weight network is padded with ones.

1.1. Network Architecture

In Fig. 1, 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 [11]. However, instead of two subsequent MLPs predicting density and color, we split our model and evaluate the weight and the offset network concurrently.

1.2. 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.

1.3. 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 T_α, for both pixels involved in the computation and their direct neighbors:

$$(\nabla\zeta)_{i,j} = (\nabla\zeta)_{i,j} \cdot \left(\prod_{y=j-1}^{j+2} (T_\alpha)_{i,y}, \prod_{z=i-1}^{i+2} (T_\alpha)_{z,j} \right). \tag{1}$$

1.4. Modifications to our Framework

We include the following modifications to torch-ngp [14] to improve reconstruction for real-world scenes, following PaletteNeRF [5]. By default, the background color is assumed to be white for real-world scenes — iNGP might cheat with this static background color by not accumulating full α along a ray. We use a random background color during training, which leads to better depth estimates and a sparser occupancy grid. In addition, we mark grid cells between each camera and its corresponding near plane as non-trainable.

2. 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.

2.1. Color Palette Regularization

To measure the fidelity of the learned color palette $\hat{\mathcal{P}}$ quantitatively, we adopt the metrics from PaletteNeRF [5] and measure sparsity with

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

In addition, we measure the total variation of the per-palette weight images $\hat{\boldsymbol{w}}_i$. We present the metrics in Tab. 1, 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. 2. 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.

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

Table 1. Quantitative Evaluation of our proposed color palette regularization. Our full model achieves the best metrics for sparsity and uses the fewest base colors.

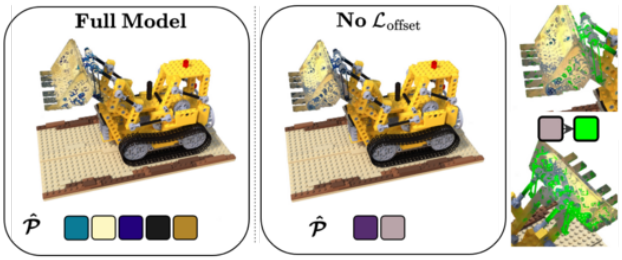

Figure 2. **Ablation study** on the effect of $\mathcal{L}_{\text{offset}}$. Without this loss term, our model suffers from incorrect palette reconstruction and highly non-intuitive recoloring.

2.2. 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. 3. As can be seen, LAENeRF retains 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 $\hat{\mathcal{P}}$.

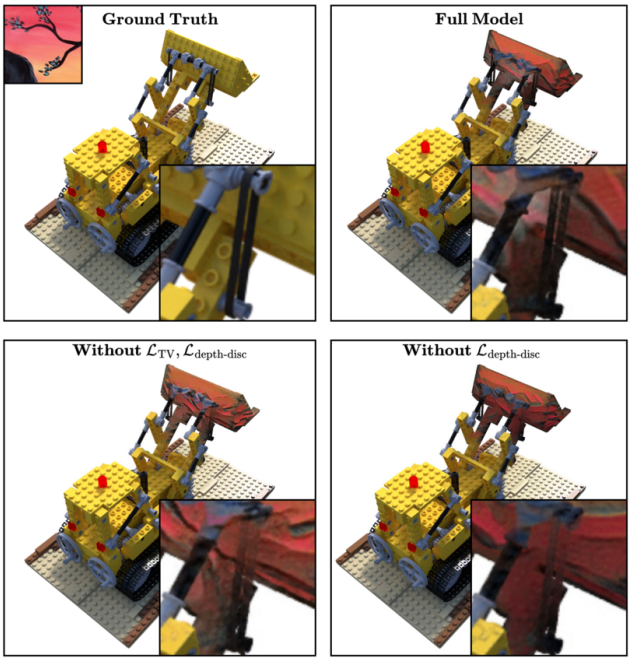

Figure 3. **Ablation study** on the effectiveness of our proposed losses for 3D-aware stylization. Our full model retains the most detail during stylization.

2.3. View-Consistency

In addition to our experiments in the main paper, we evaluate a view-consistency metric from SNeRF [12], which is a generalization of Lai *et al.* [6] to the NeRF setting. Following SNeRF, we estimate the optical flow using RAFT [15] and use the occlusion detection method from Ruder *et al.* [13] 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

$$E_{\text{warp}}(\boldsymbol{V}_{i+\delta}, \bar{\boldsymbol{V}}_{i+\delta}) = \frac{1}{|O|} \left\| \bar{\boldsymbol{V}}_{i+\delta} - \boldsymbol{V}_{i+\delta} \right\|_2^2. \tag{3}$$

In addition, we also measure LPIPS [16] between $\bar{\boldsymbol{V}}_{i+\delta}$ and $\boldsymbol{V}_{i+\delta}$, following StyleRF [8]. We show our results in Tab. 2, 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 [17] for all metrics. PaletteNeRF [5] with semantic features achieves slightly better results for MSE, but performs worse for LPIPS. These quantitative results align with the results for our user study.

	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

Table 2. **View-Consistency comparison** of our method, Ref-NPR [17] and PaletteNeRF [5] with semantic guidance. We measure short-range consistency ($\delta = 1$) and long-range consistency ($\delta = 7$).

3. 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 [9] (*Flower, Horns, Fortress, Orchids, Trex*) and the mip-NeRF 360 dataset [1] (*Kitchen, Bonsai, Room*) and compare against PaletteNeRF [5] 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 [9] (*Flower*, *Horns*, *Trex*) and the NeRF-Synthetic dataset [10] (*Lego, Hotdog, Chair, Drums*) and compare against Ref-NPR [17]. 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 [2], as shown in Fig. 4. 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 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 ($Z = 1890.0;\ p > 0.5$).

4. GUI

LAENeRF incorporates a GUI building on [3, 14] for real-time, interactive appearance editing of NeRFs. Here, we present several key components of our user interface.

4.1. Region Selection and Growing

We provide an intuitive region selection process, based on NeRFShop [3]. 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 the user controls 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. Due to the region growing procedure and easy deletion of voxels with a second grid and binary operations, our method is quite insensitive to the user scribble provided as input. The complete workflow can be seen in Fig. 5.

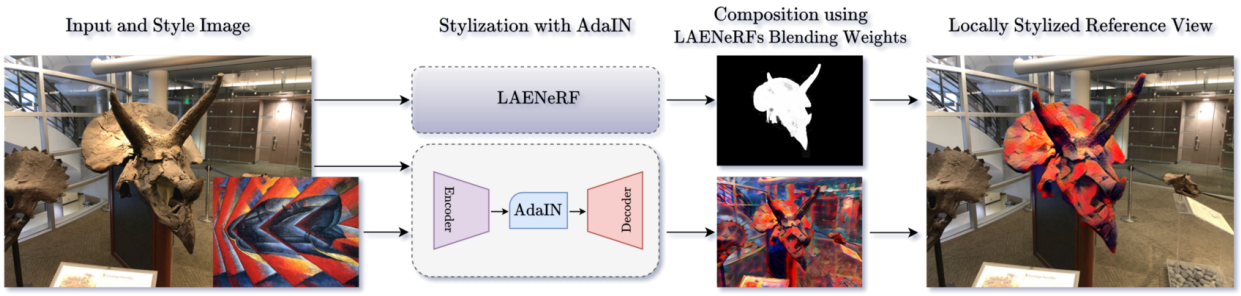

Figure 4. **Generation of locally stylized reference images** for Ref-NPR [17] using AdaIN [2] and LAENeRF's blending weights.

Figure 5. **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 $\boldsymbol{\mathcal{E}}$. Finally, this initial selection can be extended with region growing (2 iterations shown here).

4.2. Style Image Selection

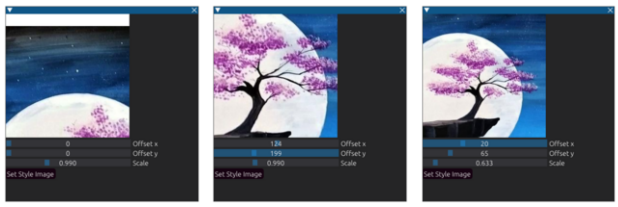

Figure 6. **GUI interface** for style image selection.

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. 6, our GUI supports zooming and cropping arbitrary images to the required size.

4.3. 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 palette weights $\boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}} \in \mathbb{R}^{N_{\hat{\boldsymbol{\mathcal{P}}}}}$ and palette biases $\boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}} \in [-1,1]^{N_{\hat{\boldsymbol{\mathcal{P}}}}}$, which transform the weights according to

$$\begin{aligned} \hat{\boldsymbol{w}} &= \min(\hat{\boldsymbol{w}} \cdot \boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}} + \boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}}, 0), \\ \hat{\boldsymbol{w}} &= \frac{\hat{\boldsymbol{w}}}{\mathbf{1}^T \hat{\boldsymbol{w}}}. \end{aligned} \tag{4}$$

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. 7.

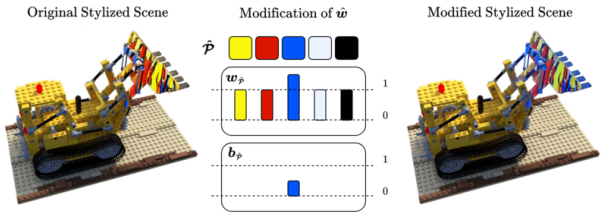

Figure 7. **Demonstration of user-guided modifications to $\hat{w}$.** We enable modification of stylized regions by changing the importance of individual base colors $\hat{\boldsymbol{\mathcal{P}}}_i$ via modification of $\boldsymbol{w}_{\hat{\boldsymbol{\mathcal{P}}}}, \boldsymbol{b}_{\hat{\boldsymbol{\mathcal{P}}}}$.

4.4. Preview Mode & GUI demonstration

To enable interactive recoloring and stylization, our approach supports real-time rendering during training of our LAENeRF module. After the training dataset has been extracted, which takes ~ 15 seconds depending on the dataset, users can watch our neural module converge. As we only handle views of the training dataset during optimization, this efficient dataset extraction results in fast convergence.

To render novel views during training, enabling interactive previews, we query Θ_{NeRF} to obtain the estimated ray termination $\boldsymbol{x}_{\text{term}}$ and T_α, as we do during training dataset extraction. If $T_\alpha > 0.5$, we use $\boldsymbol{x}_{\text{term}}$ as input to LAENeRF and output the rendered image. Consequently, due to the fast stylization process, users have the ability to stop training after few seconds and choose another style image if they find the result unappealing.

Figure 8. **Example of Color Edits** for the LLFF dataset [9] for our method and PaletteNeRF [5].

4.5. Detailed Time Comparisons

We provide a more detailed analysis of the time comparison to PaletteNeRF [5] in Tab. 3. As can be seen, LAENeRF performs the recoloring task much faster, although the same NeRF backbone is used.

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**

Table 3. **Detailed timing comparison** of our method and PaletteNeRF [5].

5. 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.

5.1. LLFF Recoloring

For the LLFF dataset [9], we show our per-recoloring results in Tab. 5. All masks were provided to us by the authors of ICE-NeRF [7]. For our color edits, we also include substantial recolorings, as can be seen in Fig. 8. PaletteNeRF [5] 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.

5.2. mip-NeRF 360 Recoloring

For the mip-NeRF 360 dataset [1], we report per-recoloring results in Tab. 6. We measure MSE in the background of the selected region, with masks extracted using Segment Anything [4] 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. 9.

Figure 9. **Foreground masks** for the mip-NeRF 360 dataset [1]. We lowered brightness and saturation and additionally blurred the background using the masks.

5.3. Local Stylization

For local stylization, we report per scene results in Tab. 4. As can be seen, our method outperforms Ref-NPR [17] for every scene.

Method	NeRF-Synthetic [10]			
	Chair	*Drums*	*Lego*	Average
Ref-NPR [17]	0.0357	0.0808	0.0233	0.0466
LAENeRF	0.0037	0.0154	0.0021	**0.0071**
	LLFF [9]			
	Horns	*Flower*	*Trex*	Average
Ref-NPR [17]	0.0075	0.0079	0.0064	0.0073
LAENeRF	0.0024	0.0026	0.0025	**0.0025**

Table 4. **MSE (↓) in the background** per scene for local stylization for our method and Ref-NPR [17].

References

[1] 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.

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

[3] 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.

[4] 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] 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.

	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 5. **MSE (↓) in the background per recoloring** for the LLFF dataset [9] for our method and PaletteNeRF [5] with and without semantic guidance.

	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**

Table 6. **MSE (↓) in the background per recoloring** for the mip-NeRF 360 dataset [1] for our method and PaletteNeRF [5] with and without semantic guidance. LAENeRF outperforms PaletteNeRF for every recoloring, demonstrating effectiveness for diverse recolorings.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic Style Transfer for Videos. In *Proceedings of the German Conference on Pattern Recognition*, 2016.

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

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

[16] 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.

[17] 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.