PaletteNeRF: Palette-based Appearance Editing of Neural Radiance Fields – Supplementary Materials

Zhengfei Kuang[1]*, Fujun Luan[2], Sai Bi[2], Zhixin Shu[2], Gordon Wetzstein[1], Kalyan Sunkavalli[2]
[1]Stanford University [2]Adobe Research
{zhengfei,gordonwz}@stanford.edu {fluan,sbi,zshu,sunkaval}@adobe.com

https://palettenerf.github.io

1. Implementation Details

1.1. Network

Fig. 1 Shows the detailed architecture of our model. Since we use Instance-NGP [6] to reconstruct the scene geometry, our decomposition network is also based on Instance-NGP. It consists of two stages: The first branch is inherited from the vanilla Instant-NGP's geometry network, generating the density σ and a geometry feature z_σ. The second stage takes the geometry feature z_σ, the position $\boldsymbol{x}$, and the viewing direction $\boldsymbol{d}$ as input, and outputs the palette-based basis functions and the view-dependent color. We use an alternative hashing-based encoder in this branch, since the original encoder is frozen to keep the pre-trained geometry unchanged. Furthermore, for semantic-guided recoloring, the model can be further extended with a semantic network, which has an identical structure to the geometry network and generates the semantic feature only.

All our networks are implemented based on the PyTorch version of Instant-NGP written by Tang et al. [8].

1.2. Training of Decomposition Model

As mentioned in the paper, we calculate the palette blending weights of image pixels using the method from Tan et al. [7] as supervision. While the original method employs spatial information of the pixels (i.e., coordinates), we do not use them for view consistency issues. To speed up, we build a $32\times32\times32$ grid on the RGB space and calculate the weights on the grid nodes, then interpolate the weights to all image pixels.

We set $\lambda_\text{s}, \lambda_\text{sp}, \lambda_\text{offset}, \lambda_\text{sm}, \lambda_\text{palette}, \lambda_\text{weight}$ to 0.1, 0.0002, 0.03, 0.004, 0.001 and 0.05 in all of our experiments. We load the geometry model from the vanilla Instant-NGP, and freeze the parameters during the training. For the first 100 epochs, We freeze the palette weights, and linearly decrease λ_weight per epoch to 0.

If the model contains the semantic network, we also add two losses on the predicted feature maps: a reconstruction loss defined as the L2 distance to the ground truth maps, and a smooth loss which is similarly defined as $\mathcal{L}_\text{sm}$. They share the same weights with $\mathcal{L}_\text{recon}$ and $\mathcal{L}_\text{sm}$.

1.3. Recoloring

Given novel palettes, we first calculate their changes to the original palettes in the HSV space. Specifically, we calculate the difference between the H value, and the scale of the S and V values. We then copy these changes to the per-point soft color $\boldsymbol{c}_p \in [0,1]^{N_p\times3}$ (i.e., palette color add color offset) to get $\boldsymbol{c}'_p$. If no semantic guidance is employed, the model directly outputs $\boldsymbol{c}'_p$. Otherwise, it outputs:

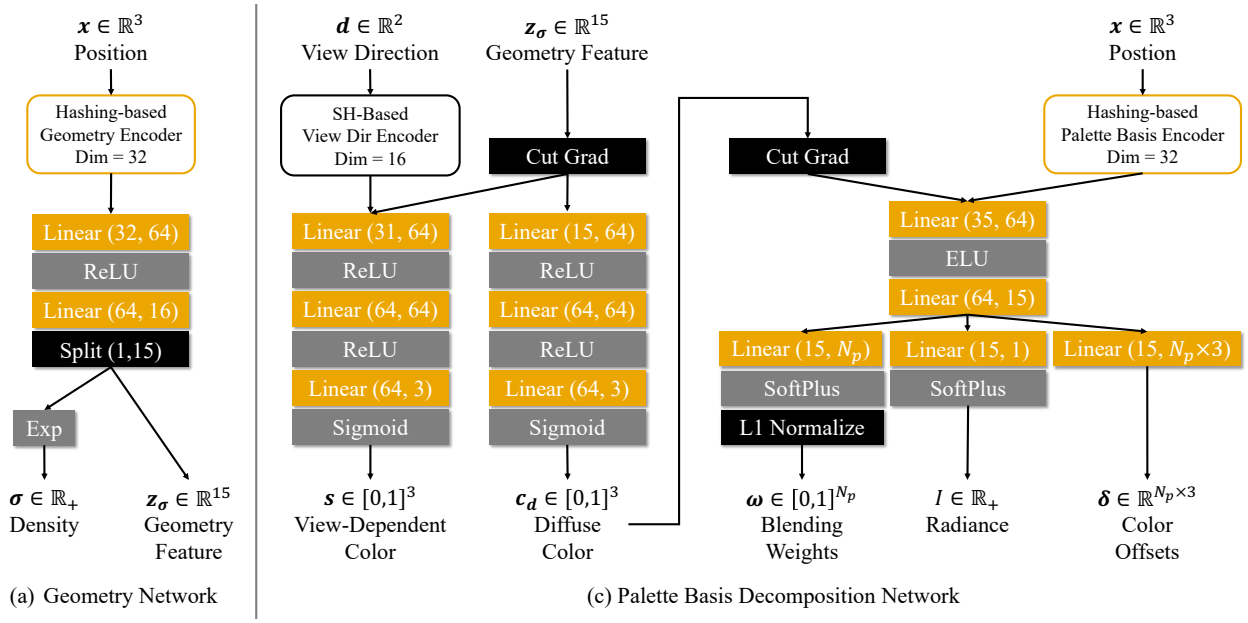
$$\text{Lerp}\left(\boldsymbol{c}_p, \boldsymbol{c}'_p, \exp(-\frac{\|\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{\mu}_{\boldsymbol{f}}\|^2}{\sigma_{\boldsymbol{f}}})\right), \tag{1}$$

where $\boldsymbol{f}$ is the learned feature field, $\boldsymbol{\mu}_{\boldsymbol{f}}$ and $\sigma_{\boldsymbol{f}}$ are the mean value and variance of the semantic feature controlled by the user. In practice, the user can select the mean value and variance through our GUI, which we also show in our supplementary video.

Photorealistic Style Transfer Given a set of correspondences of 3D points and style image pixels, our model supports photorealistic style transfer by optimizing a transformation of the basis functions. Specifically, the transformation is composed of three types of parameters: The difference of the radiance function $dI \in \mathbb{R}^{N_p}$ (independent for each basis); The difference of the color palettes $d\boldsymbol{\mathcal{P}} \in \mathbb{R}^{N_p\times3}$; And the transformation matrices of the color offsets $R^{\boldsymbol{\delta}} \in \mathbb{R}^{N_p\times3\times3}$. With them, we modify the color composition equation in the paper (Eq. [2]) to:

$$\boldsymbol{c}(\boldsymbol{x},\boldsymbol{d}) = \boldsymbol{s}(\boldsymbol{x},\boldsymbol{d}) + \sum_{i=1}^{N_p} I'_i(\boldsymbol{x})\omega_i(\boldsymbol{x})\left(\boldsymbol{\mathcal{P}}'_i + \boldsymbol{\delta}'_i(\boldsymbol{x})\right), \tag{2}$$

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



(a) Geometry Network

(c) Palette Basis Decomposition Network

Figure 1. **Network structure of our model**. Components painted with yellow color have trainable parameters. The geometry network's parameters are loaded from the pre-trained model and fixed during the training. For details of the encoders, please refer to the original Instant-NGP [6] paper.

where $\mathcal{P}'_i = \mathcal{P}_i + d\mathcal{P}_i, \boldsymbol{\delta}'_i(\boldsymbol{x}) = R_i^{\boldsymbol{\delta}} \boldsymbol{\delta}_i(\boldsymbol{x}), I'_i(\boldsymbol{x}) = I(\boldsymbol{x}) + dI_i$. Two losses are applied for the optimization: a reconstruction loss defined as the sum of L2 distance between the updated diffuse colors (i.e., the sum of all basis colors) and the corresponding style colors of all point-pixel correspondences, and a regularization loss defined as:

$$\mathcal{L}_{\text{reg}} = \sum_i^{N_p} \|d\mathcal{P}_i\|^2 + \|R_i^{\boldsymbol{\delta}} (R_i^{\boldsymbol{\delta}})^T - I\|^2. \tag{3}$$

These two losses are weighted by 1 and 0.1, respectively. We optimize the parameters for 1000 steps using an Adam optimizer with a learning rate of 0.001.

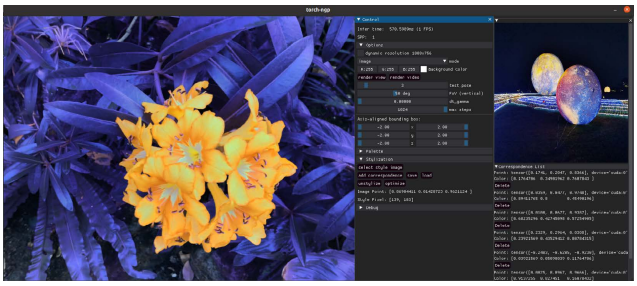


Figure 2. **Photorealistic style transfer with our GUI**. Our system can optimize a stylized scene given a style image (right top) and several point-pixel correspondences (right bottom).

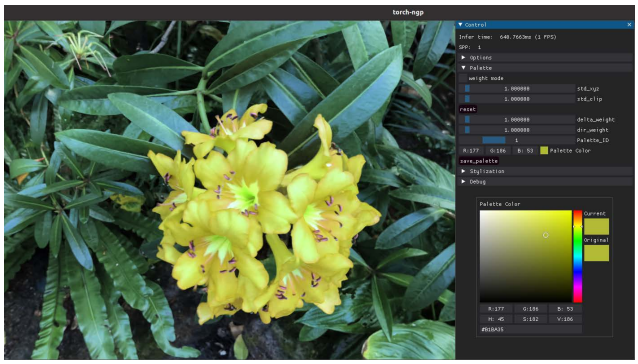


Figure 3. **Recoloring with our GUI**. Our GUI allows users to select and modify palettes (right bottom) for real-time recoloring.

1.4. Interactive GUI

Our GUI is implemented on the framework of DearPyGUI [3]. It supports various applications, including novel view synthesis, intuitive recoloring (with semantic guidance), photorealistic style transfer, and our additional appearance editings (e.g., illumination changing). Fig. 3 and Fig. 2 show two examples of our GUI, and we also provide demos in our supplementary video.

2. Details of User Study

We conduct our user study on the task of palette-based recoloring and photorealistic style transfer. For recoloring, we select ten scenes from the LLFF dataset [5], Blender Dataset [4] and the MipNeRF-360 Dataset [1], and render

two views per scene. We manually edit the palettes per model per view with the same recoloring target (e.g., changing the flower's color to yellow) without limiting the number of changed palettes. For style transfer, we select four scenes from the LLFF dataset and assign two style images for each (8 pairs in total). For each pair, we run all methods to render a stylized video with the same trajectory.

We compare our method with all baselines for each task using photorealism questions and view-consistency questions. In each photorealism question, users are provided with a reference image (video), an image (video) from our method and an image (video) from one baseline, and decide which result is more photorealistic. In each view-consistency question, users are given two views (a video) generated from our method and two views (a video) generated from another baseline, and decide which results are more view-consistent.

3. Additional Experiments

In addition to the evaluations in our paper, we show more experiment results in this section.

3.1. Comparison with Du et al. [2]

We show another qualitative comparison between our model and the state-of-the-art palette-based video recoloring method of Du et al. [2]. As shown in Fig 4, the method of Du et al. performs decently when only editing the plant and table, but introduces view-inconsistency when editing the floor and bicycle, too. On the other hand, our results are view-consistent with all edits. We also show the animated comparison in our supplementary video.

3.2. Ablation Study on Basis Decomposition

As the supplement of the ablation study in the paper, we show more qualitative results of the basis color maps in Fig 7. Our full model achieves the most sparse and 3D reasonable decomposition results compared to other variants.

3.3. Ablation Study on Semantic Feature Compression

To find the most appropriate dimension of the compressed semantic feature, we evaluate the effects of different dimensions with three metrics: our model's inference time, our model's training memory cost, and a novel feature distance error $\mathcal{E}_{\text{FD}}$. The feature distance error is given by:

$$\mathcal{E}_{\text{FD}} = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}} \left(\frac{\left| \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\|^2 + \|\boldsymbol{f}'(\boldsymbol{x}) - \boldsymbol{f}'(\boldsymbol{y})\|^2 \right|}{\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\|^2} \right), \quad (4)$$

where $\boldsymbol{f}$ and $\boldsymbol{f}'$ are the original and compressed semantic feature maps, and $\boldsymbol{x}, \boldsymbol{y}$ is a pair of pixels. This error calculates the fidelity of the compressed feature. We sample pixel pairs across the training images multiple times in our experiment to approximate this error. We test various dimensions on the MipNeRF-360 datasets, and the results are shown in Fig. 5. Choosing 16 as the feature dimension only brings minor effects on the features' fidelity and the model's efficiency. Choosing 32 as the dimension, however, can cause an out-of-memory problem with the preload setting (a common optimization that load all training images to the GPU before training). Thus we use the dimension of 16 in all experiments.

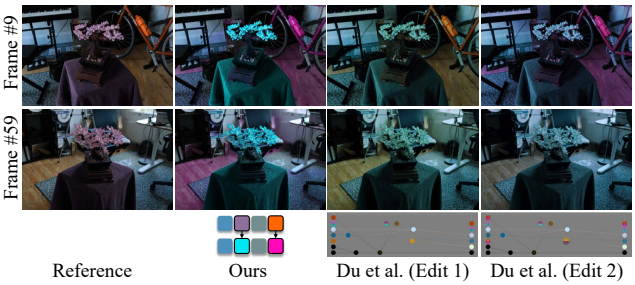


Figure 4. **Comparison with Du et al. [2].**

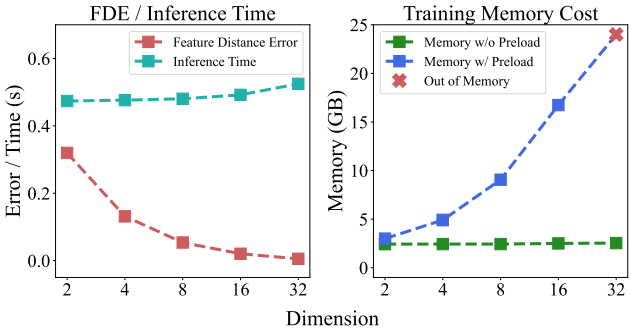


Figure 5. **Ablation study on the semantic feature.**

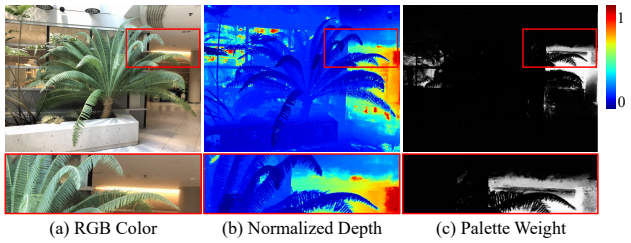


Figure 6. **Example of our model's failure case**. While the color of the ceiling is smooth in the image space, our model is affected by the discontinuous geometry reconstructed by the Instant-NGP and outputs weight maps with a rough boundary in the highlighted area.

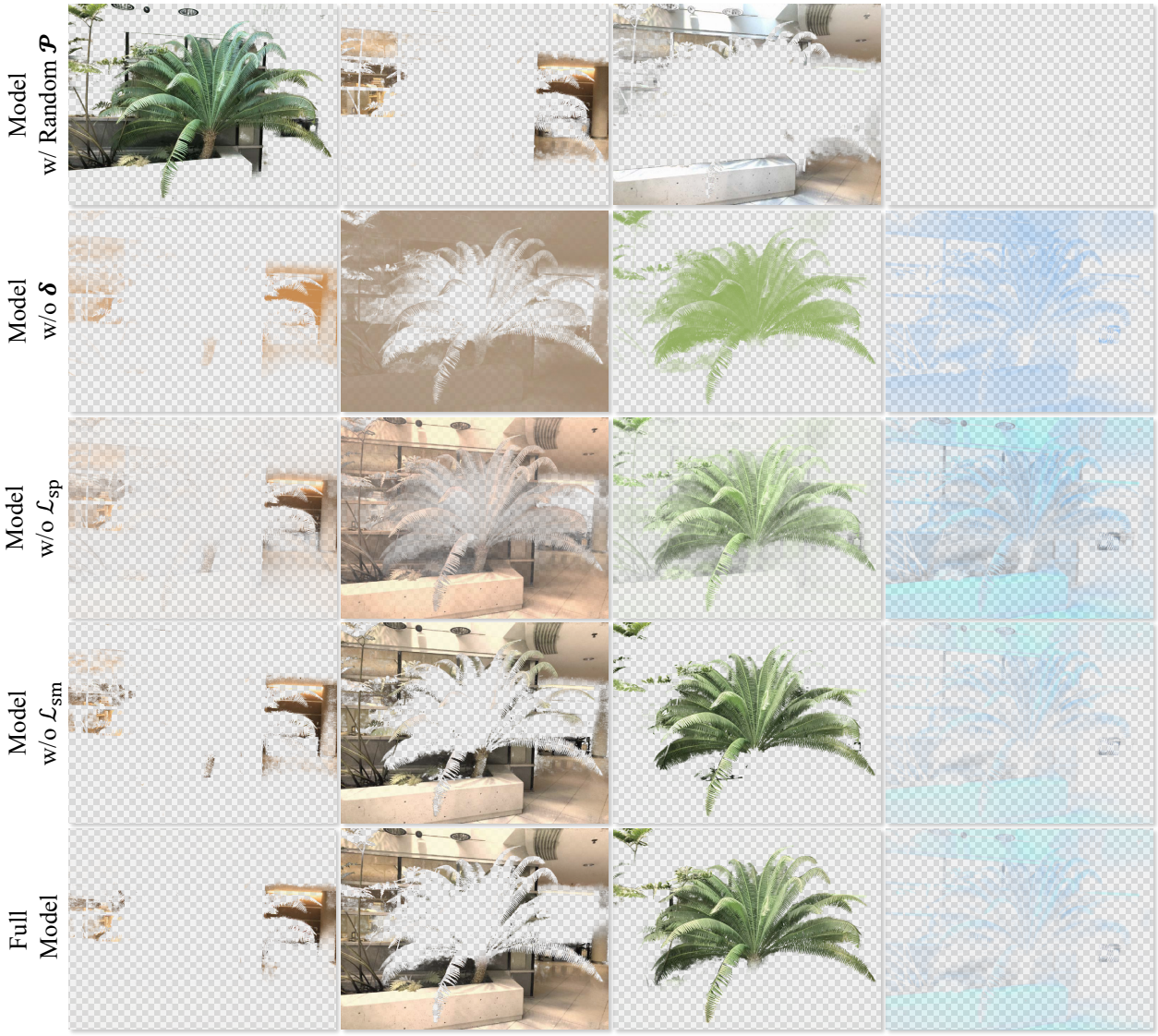


Figure 7. **Ablative study results on basis color maps**. Our model trained with randomly initialized palettes (model w/ Random $\mathcal{P}$) generates bases with inordinate color offsets. Our model trained without color offset(Model w/o δ, and model trained without sparsity loss (Model w/o $\mathcal{L}_{sp}$) predicts decomposition results with less sparsity. Our model trained without the smooth loss (Model w/o $\mathcal{L}_{sm}$) generates rough segmentation.

4. Limitation

While we carefully design our model to produce clean and 3D consistent decomposition results, it may produce rough segmentation in some cases due to the reconstruction failure of our geometry network. Fig 6 shows an example of this situation. We believe that our method can be further improved when built with better geometry.

5. Additional Results

Finally, we show additional qualitative results of our model. We provide more recoloring results in Fig. 8, Fig. 9 and Fig. 10, and appearance editing results in Fig. 11 and Fig. 12 . Please refer to our supplementary video for more animated results.

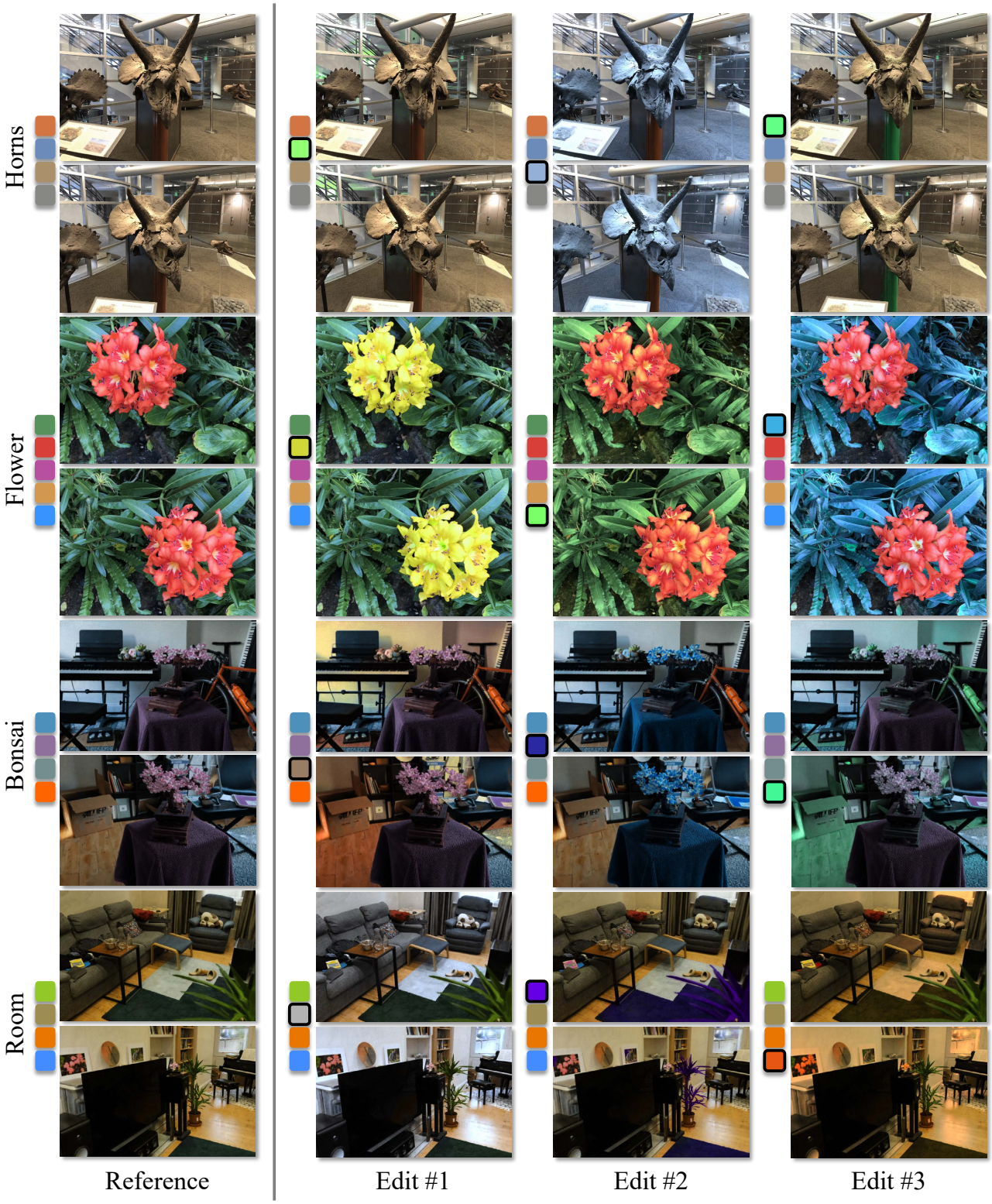


Figure 8. **Recoloring results on real datasets.**

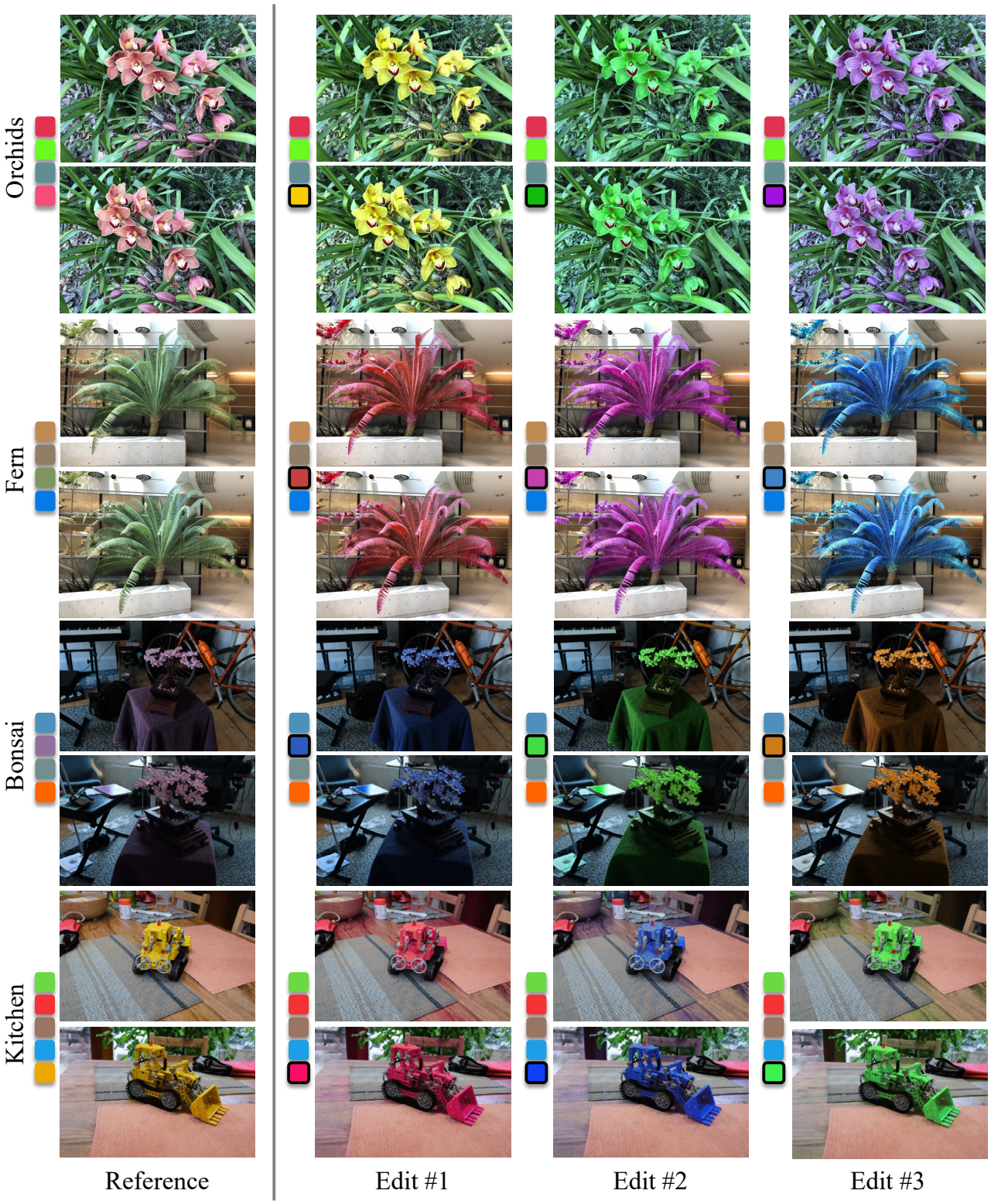


Figure 9. **Recoloring results on real datasets.**

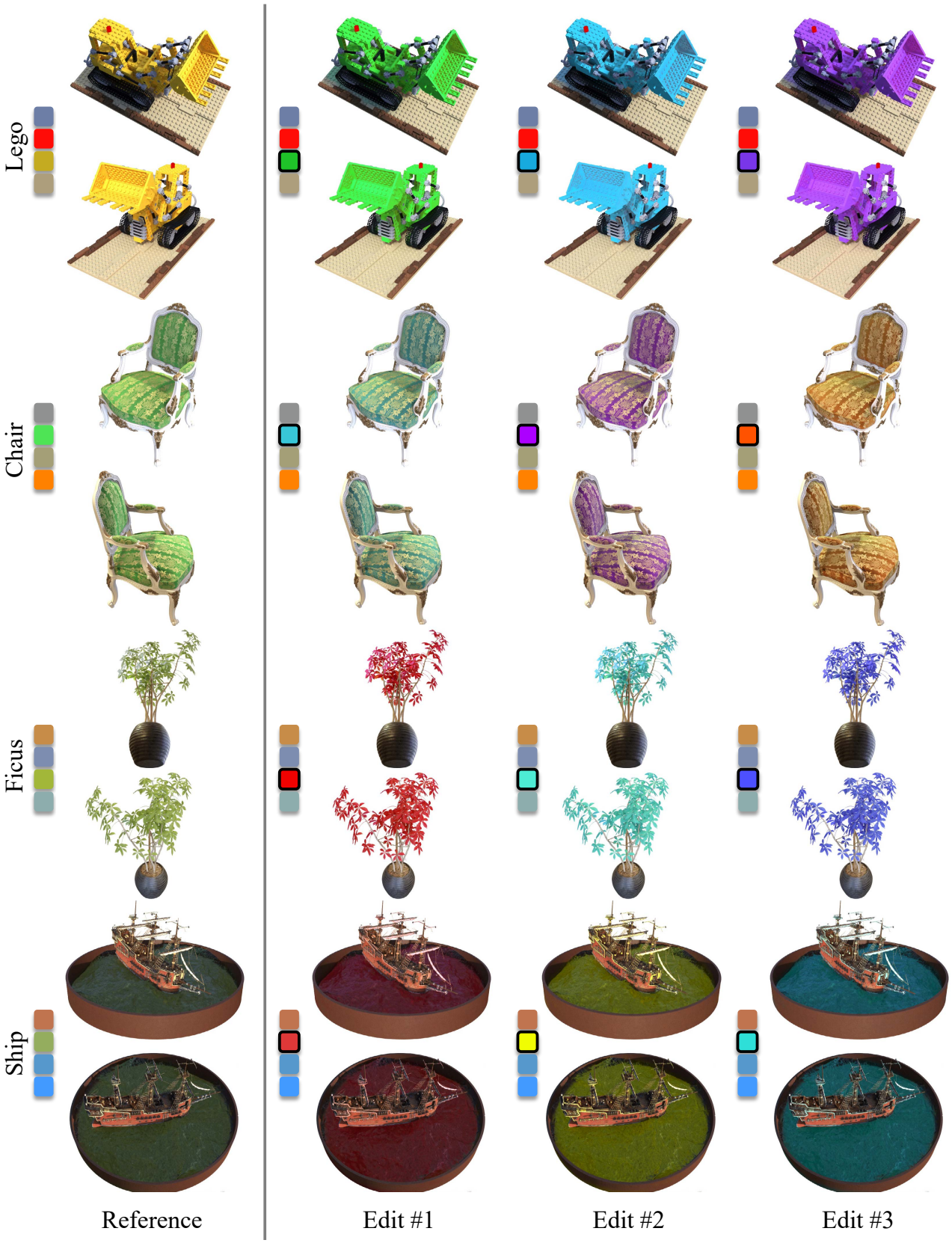


Figure 10. **Recoloring results on synthetic datasets.**

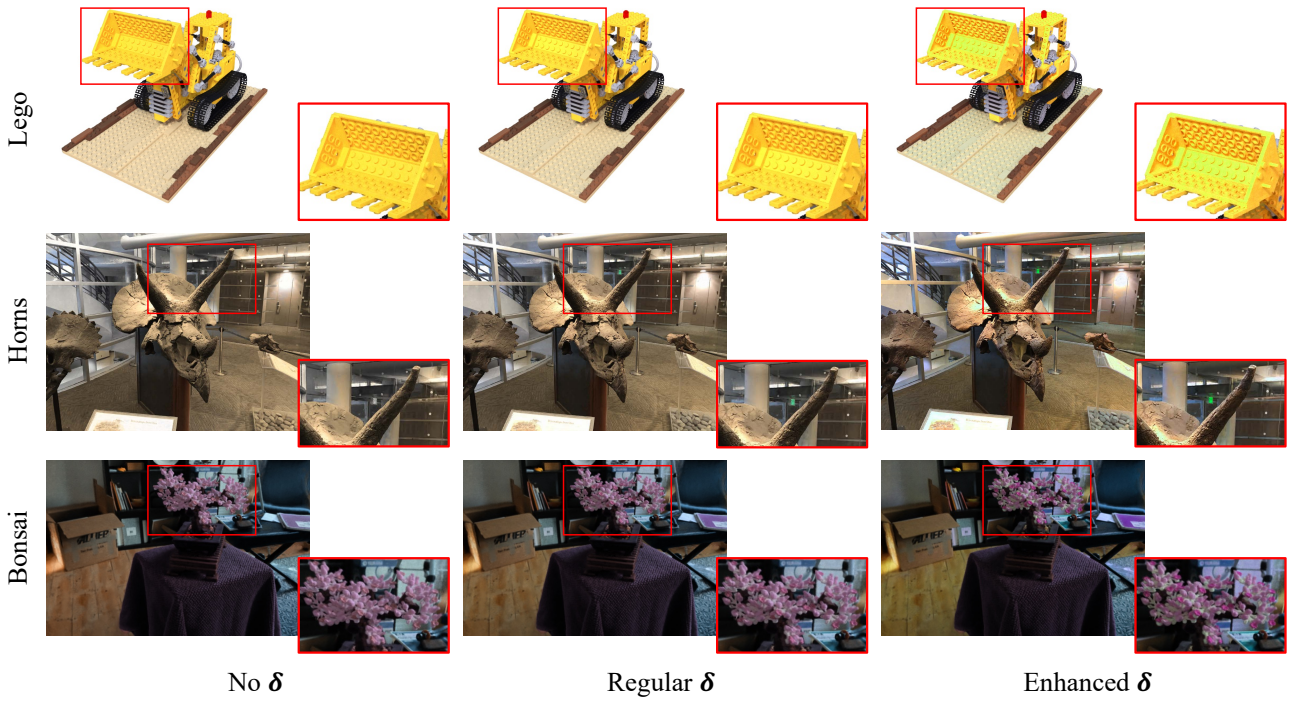


Figure 11. **Qualitative results of color offset editing**. For each scene, we show editing results where the color offset is removed (No δ), unchanged (Regular δ), and enhanced (Enhanced δ).

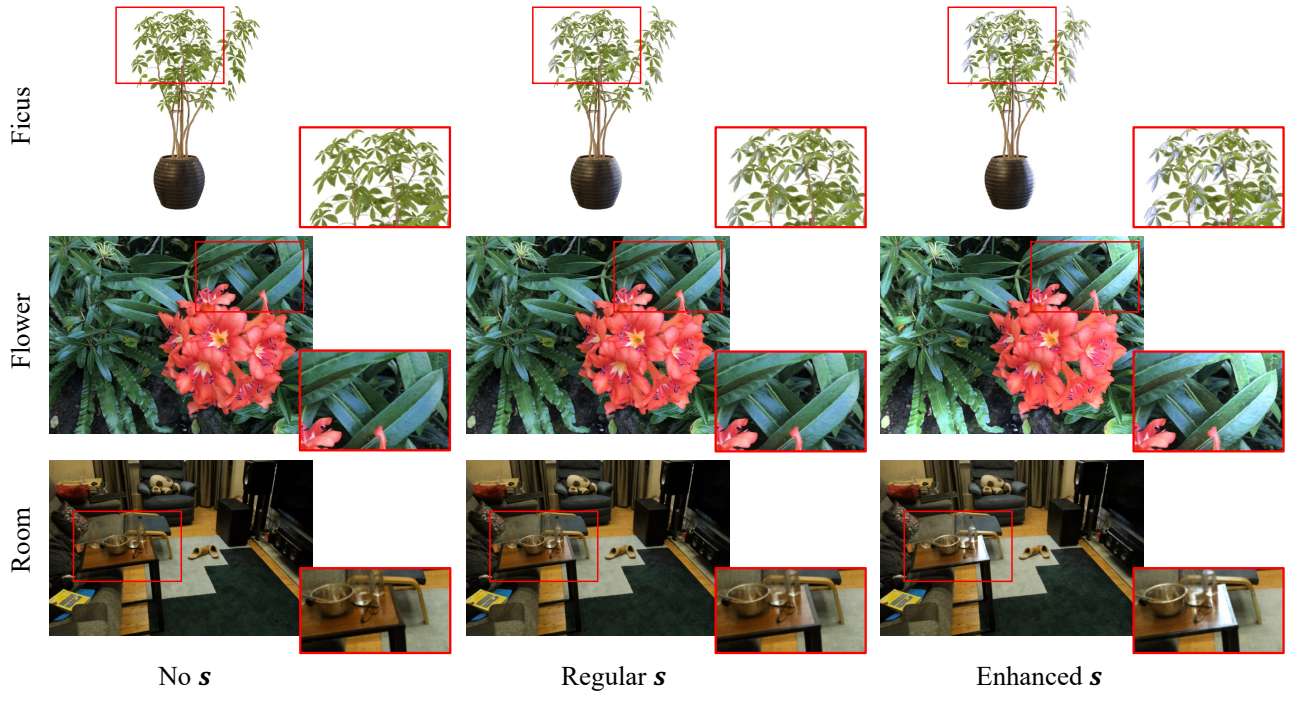


Figure 12. **Qualitative results of view-dependent color editing**. For each scene, we show editing results where the view-dependent color is removed (No s), unchanged (Regular s), and enhanced (Enhanced s).

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2

[2] Zheng-Jun Du, Kai-Xiang Lei, Kun Xu, Jianchao Tan, and Yotam Gingold. Video recoloring via spatial-temporal geometric palettes. *ACM Transactions on Graphics (TOG)*, 40(4), Aug. 2021. 3

[3] Jonathan Hoffstadt and Preston Cothren, 2021. https://github.com/hoffstadt/DearPyGui. 2

[4] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, 2022. 2

[5] Pooneh Mohaghegh, Rabia Saeed, François Tièche, Alexis Boegli, and Yves Perriard. Depth camera and electromagnetic field localization system for iot application: High level, lightweight data fusion. In *ASSE 2021: 2nd Asia Service Sciences and Software Engineering Conference, Macau, 24-26 February, 2021*, pages 94–101. ACM, 2021. 2

[6] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[7] Jianchao Tan, Jose Echevarria, and Yotam Gingold. Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. *ACM Transactions on Graphics (TOG)*, 37(6):262:1–262:10, Dec. 2018. 1

[8] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 1