Article Type: Description (see Introduction for more detail)

NeRF-In: Free-Form Inpainting for Pretrained NeRF with RGB-D Priors

I-Chao Shen*, *The University of Tokyo, Japan*
Hao-Kang Liu*, *National Taiwan University, Taiwan*
Bing-Yu Chen, *National Taiwan University, Taiwan*

Abstract—Neural Radiance Field (NeRF) has emerged as a versatile scene representation. However, it is still unintuitive to edit a pretrained NeRF because the network parameters and the scene appearance are often not explicitly associated. In this paper, we introduce the first framework that enables users to retouch undesired regions in ***a pretrained NeRF scene*** *without accessing any training data and category-specific data prior. The user first draws a free-form mask to specify a region containing the unwanted objects over an arbitrary rendered view from the pretrained NeRF. Our framework transfers the user-drawn mask to other rendered views and estimates guiding color and depth images within transferred masked regions. Next, we formulate an optimization problem that jointly inpaints the image content in all masked regions by updating NeRF's parameters. We demonstrate our framework on diverse scenes and show it obtained visually plausible and structurally consistent results using less user manual efforts.*

Introduction

Recent advancements in neural rendering, such as Neural Radiance Fields (NeRF) [1], has emerged as a powerful representation for the task of novel view synthesis, where the goal is to render unseen viewpoints of a scene from a given set of input images. NeRF encodes the volumetric density and color of a scene within the weights of a coordinate-based multi-layer perceptron. Several follow-up works extend original NeRF to handle different tasks, such as pose estimation [2], deformable 3D reconstruction [3], and modeling dynamic scenes [4].

Though NeRF achieves great performance on photo-realistic scene reconstruction and novel view synthesis, there remains enormous challenges in editing the geometries and appearances of a scene represented by a pretrained NeRF model. First, a user needs to edit on multiple rendered views to edit the whole scene which is often challenging because of the ambiguous correspondences of the edited regions across multiple views. Second, because millions of parameters are used in a pretrained NeRF model, it is unclear which parameters control the different aspects of the rendered scene and how to change the parameters according to the sparse local user input. There are some early attempts on enabling users to perform object-level [5] or attribute-level [6] editing over NeRF models. However, these methods require additional category-level or object-level conditioned training and additional data to support the desired editings.

In this paper, we focus on the free-form inpainting problem on a pretrained NeRF model, *i.e.*, removing unwanted regions or objects using a free-form mask in a 3D scene represented by a pretrained NeRF. Although we can ask a user to provide a mask and the inpainted image for each rendered view, and use them for training a new NeRF, there are several disadvantages. First, it is labor-intensive to provide masks for every rendered view. Second, there will be visual inconsistency across different inpainted views introduced by separate inpaintings.

To address these issues, we propose a framework to assist users in removing unwanted regions or objects by updating a pretrained NeRF. Given a

Digital Object Identifier 10.1109/XXX.0000.0000000
*Both authors contributed equally to the paper.

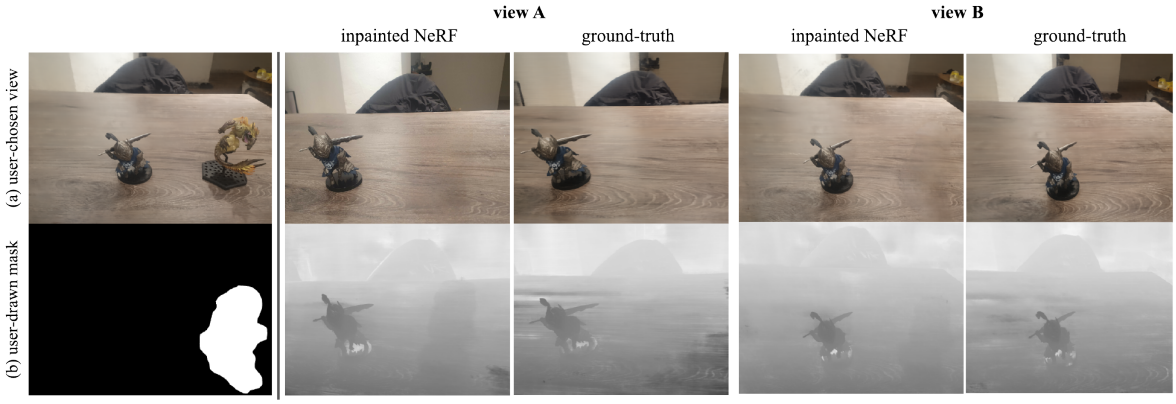

FIGURE 1: Given **a pretrained NeRF model**, a user can (a) choose an arbitrary view and (b) draw a free-form mask to specify the unwanted region or object in the 3D scene. Our framework optimizes the parameters of the pretrained NeRF based on the user-drawn mask and removes the unwanted object in the masked region. The optimized NeRF synthesizes the inpainted results that match the ground truth from different views. (Noted that the user-chosen view is different from view A and view B.) Please refer to the supplementary videos for video results.

pretrained NeRF, a user first draws a free-form mask to specify the unwanted region over an arbitrary rendered view. Our framework then renders a couple of views sampled from the pretrained NeRF based on a preset trajectory, transfers the user-drawn mask to these sampled views using a video object segmentation method [7], and generates (i) guiding color images using [8] and (ii) guiding depth images using Bilateral Solver [9] within these masked regions. Noted that our framework is agnostic to the methods chosen, *i.e.*, we can use any existing method for transferring the mask and generating the guiding color and depth images. Finally, we formulate an optimization problem that jointly inpaints the image content within the transferred masked regions with respect to the guiding color and depth images. Our optimization formulation inpaint a pretrained NeRF model using any NeRF architectures, unlike concurrent NeRF inpainting works [10], [11] require to train additional components.

We demonstrate our framework on several scenes represented by pretrained NeRFs in LLFF dataset [12], and show that our framework generates visually plausible and consistent results. We also demonstrate our experiments on a custom dataset to show the correctness between inpainted results and ground truth.

To sum up, we make the following contributions:

- we present the first framework that enables users to retouch undesired regions in a pretrained NeRF scene without accessing any training data and category-specific data prior.
- we gather a custom dataset with ground truth inpainting results to quantitatively evaluate the NeRF inpainting results.

Related Work

Novel view synthesis

Constructing novel views of a scene captured by multiple images is a long-standing problem in computer graphics and computer vision. Neural Radiance Fields (NeRF) [1] uses a multi-layer perceptron (MLP) and positional encoding to model a radiance field at an unprecedented level of fidelity.

These recent advances greatly improve the practical use of NeRF. However, it is still unintuitive how a user can edit a pretrained NeRF model. The main reason is that the neural network of a NeRF model has millions of parameters. Which parameters control the different aspects of the rendered shape and how to change the parameters to achieve desired edits are still unknown. Previous works enable users to select certain object [13], duplicate and move object [14], [15] and edit a NeRF model using strokes [5] directly. However, these methods require learning additional category-level or object-level conditional radiance fields to facilitate such edits. Unlike these methods, our framework requires neither additional category-specific training data nor training procedures for removing unwanted objects or regions in a pretrained NeRF model. Our method and concurrent NeRF inpainting works [10], [11] require user-provided masks while assuming different input amounts. Mirzaei *et al.* [10] take the captured RGB image sequence and sparse user annotations on a

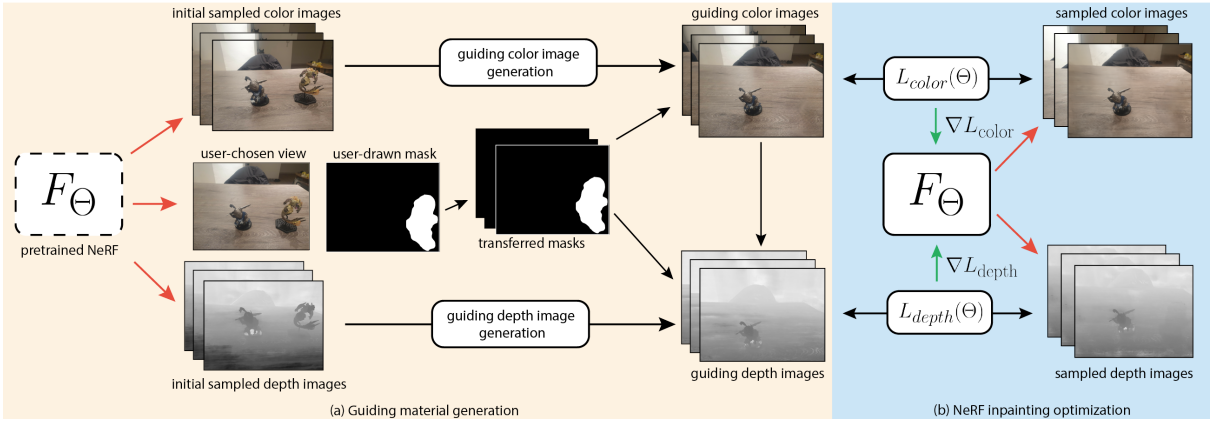

FIGURE 2: (a) Given a pretrained NeRF F_Θ, a user specifies an unwanted region on a user-chosen view with a user-drawn mask. Our framework then sampled initial color images and initial depth images from the pretrained NeRF F_Θ, and generates both guiding color images and guiding depth images. (b) Our framework updates Θ (the NeRF model parameters) by optimizing both color-guiding loss (L_{color}) and depth-guiding loss (L_{depth}). ($\rightarrow$ denotes rendering a view from F_Θ and $\rightarrow$ denotes updating Θ by optimizing the losses.)

single view. Weder *et al.* [11] take the captured RGB-D image sequence and per-frame masks of the object to be removed. Unlike these methods, our method focuses on inpainting a pretrained NeRF directly without accessing the training image sequences, and no need for per-frame object masks. Although the transferred masks may not always be of perfect quality, our method can use manually-annotated masks as input instead.

Image inpainting

In recent years, two broad approaches to image inpainting have existed. Patch-based method [16] fills the holes by searching for patches with similar low-level image features such as RGB values. The search space can be the non-hole region of the input image or from other reference images. The inpainted results are obtained by a global optimization after the relevant patches are retrieved. These methods often fail to handle large holes where the color and texture variance is high. Meanwhile, these methods often cannot make semantically aware patch selections. Deep learning-based methods often predict the pixel values inside masks directly in a semantic-aware fashion. Thus they can synthesize more visually plausible contents, especially for images like faces [17] and natural scenes [18]. However, these methods often focus on regular masks only. To handle irregular masks, researchers have proposed innovative solutions, such as partial convolution [19], which involves the masking and re-normalized of convolutions to only utilize valid pixels. Another promising approach is LaMa [20], which is a novel inpainting network that can effectively inpaint irregular regions using fast Fourier convolutions. MST inpainting [8] further considers both edge and line structure to synthesize more reasonable results. In this work, we use MST inpainting network to obtain the guiding inpainted results because of its superior performance on inpainting images while preserving structure. Noted that our framework can replace the MST inpainting with other inpainting methods since we only used the inpainted results as a guiding signal for our optimization problem.

Method

In this section, we first summarize the mechanism of NeRF [1] and then formulate our problem setting.

Preliminaries: Neural Radiance Field (NeRF)

NeRF is a continuous volumetric radiance field $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ represented by a MLP network with Θ as its weights. It takes a 3D position $\mathbf{x} = (x, y, z)$ and a 2D viewing direction $\mathbf{d} = (\theta, \phi)$ as input and outputs volume density σ and directional emitted color $\mathbf{c}$. NeRF renders the color of each camera ray passing through the scene by computing the volume rendering integral using numerical quadrature. The expected color $\hat{C}(\mathbf{r})$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ from the camera position $\mathbf{o}$ to the viewing direction $\mathbf{d}$ is defined as:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T(t_i)(1 - \exp(-\sigma(t_i)\delta_i))c(t_i), \quad (1)$$

$$\text{where} \quad T(t_i) = \exp\left(-\sum_{j=1}^{i-1} \sigma(t_j)\delta_j\right), \quad (2)$$

N denotes the total quadrature points sampled between the near plane t_n and far plane t_f of the camera, and $\delta_i = t_{i+1} - t_i$ is the distance between two adjacent points. We denote the color and density at point t_i produced by NeRF model F_Θ as $c(t_i)$ and $\sigma(t_i)$, respectively.

Using the above differentiable rendering equation, we can propagate the errors and update Θ through mean square error:

$$\mathcal{L}_{mse} = \sum_{\mathbf{r}\in\mathcal{R}} \|(\hat{C}^c(\mathbf{r}) - C(\mathbf{r}))\|_2^2 + \|(\hat{C}^f(\mathbf{r}) - C(\mathbf{r}))\|_2^2, \quad (3)$$

where $\mathcal{R}$ is a ray batch, $C(\mathbf{r}), \hat{C}^c(\mathbf{r}), \hat{C}^f(\mathbf{r})$ are the ground truth, coarse volume predicted, and fine volume predicted colors for ray $\mathbf{r}$, respectively. Beside the predicted color, the estimated depth for ray $\mathbf{r}$ is defined as $\hat{D}(\mathbf{r}) = \sum_{i=1}^{N} T(t_i)(1 - \exp(-\sigma(t_i)\delta_i))t_i$. For simplicity, we further define $F_\Theta^{\text{image}} : \mathbf{o} \rightarrow I$ and $F_\Theta^{\text{depth}} : \mathbf{o} \rightarrow D$ as functions that take a camera position $\mathbf{o}$ as input, and outputs the rendered color image I and estimated depth image D of a pretrained NeRF model F_Θ.

NeRF-In overview

Given a pretrained NeRF model F_Θ, a user can specify an unwanted region by drawing a mask M_u over a user-chosen rendered view $I_u = F_\Theta^{\text{image}}(\mathbf{o}_u)$, where $M_u = \{1, 0\}$ for pixels outside (1) or inside (0) the masked region, and $\mathbf{o}_u$ is the user-chosen camera position. Our goal is to obtain an updated NeRF $F_{\tilde{\Theta}}$ such that the unwanted region masked by M_u is removed in every rendered view. As shown in Figure 2, our method first samples K camera positions $\mathbf{O} = \{\mathbf{o}_s | s = 1...K\}$ along the test-set camera trajectory used in LLFF [12]. We set $K = 24$ throughout the results we show in this paper. For each camera position $\mathbf{o}_s$, we rendered a color image I_s and a depth image D_s using F_Θ and obtained all rendered views $\mathbf{I} = \{I_s | s = 1...K\}$ and their depth images $\mathbf{D} = \{D_s | s = 1...K\}$. One can potentially remove the unwanted region specified by M_u using the following naive method. First, the contents within the transferred masked regions on all sampled rendered views are removed. Then, Θ is updated using only the image content outside all transferred masked regions by optimizing the "masked mse (mmse)" function modified from Equation 3:

$$L_{\text{mmse}} = \sum_{\mathbf{r}\in\mathcal{R}} \|(\hat{C}^c(\mathbf{r}) - C(\mathbf{r})) \odot M(\mathbf{r})\|_2^2 + \|(\hat{C}^f(\mathbf{r}) - C_i(\mathbf{r})) \odot M(\mathbf{r})\|_2^2, \quad (4)$$

where $M(\mathbf{r})$ is the mask on the view where the ray $\mathbf{r}$ is sampled from, which can be transferred from the user-drawn mask M_u or drawn by the user directly on all views. However, because there is no explicit guidance on what image content and structure should be in the masked region, the unwanted region will remain in the result of optimizing Equation 4.

To address this issue, our method intend to provide explicit guidance of the appearance and the structure in the masked region during the update of the original NeRF. As preprocessing, our method takes the user-drawn mask M_u, sampled rendered views $\mathbf{I}$ and their depth images $\mathbf{D}$ as input, and outputs following guiding materials:

- user-chosen guiding color and depth images: I_u^G and D_u^G;
- transferred masks: $\mathbf{M} = \{M_s | s = 1...K\}$;
- guiding sampled color and depth images: $\mathbf{I}^G = \{I_s^G | s = 1...K\}$ and $\mathbf{D}^G = \{D_s^G | s = 1...K\}$.

Using these guiding materials, our method obtains updated parameters $\tilde{\Theta}$ by optimizing our NeRF inpainting (NeRF-In) formulation: $\Omega(M_u, I_u^G, \mathbf{M}, \mathbf{D}^G)$.

Guiding material generation

To generate the guiding materials for optimizing the NeRF-In formulation, the user-drawn mask M_u should be first transferred to be M_s that covers the same region as M_u for each sampled rendered view I_s (where a video object segmentation method (STCN) [7] is used). Note that we apply the video object segmentation method on the images generated by a pretrained NeRF, not the captured images; thus, these images can be considered as close frames. With the transferred masks M_s, we need to generate the guiding color and depth images I_s^G and D_s^G. The guiding color image generation can be described as

$$I_s^G = \rho(I_s, M_s), \quad (5)$$

where ρ is a single image inpainting method (where the MST inpainting network [8] is used). After obtaining I_s^G, we can obtain the guiding depth image using

$$D_s^G = \tau(D_s, M_s, I_s^G), \quad (6)$$

where τ is a depth image completion method (where the Fast Bilateral solver [9] is used). For the user-chosen rendered view I_u, *i.e.*, where the user drew the mask M_u, we generate the user-chosen guiding color and depth images I_u^G and D_u^G using Equation 5 and Equation 6, respectively. Noted that our framework can replace ρ to any other single image inpainting method and τ to any other single depth image completion method.

NeRF inpainting optimization

We obtain the updated parameters $\tilde{\Theta}$ that removes the unwanted region in the 3D scene by optimizing

$$\tilde{\Theta} \coloneqq \arg\min_{\Theta} \left(L_{\text{color}}(\Theta) + L_{\text{depth}}(\Theta)\right). \tag{7}$$

Color-guiding loss The color-guiding loss is defined as

$$L_{\text{color}}(\Theta) = \underbrace{\sum_{\mathbf{o}\in\mathbf{O}^{\text{in}}} l_{\text{color}}(\mathbf{o}, \Theta, (1 - M_{\mathbf{o}}))}_{\text{inside the mask}} + \underbrace{\sum_{\mathbf{o}\in\mathbf{O}} l_{\text{color}}(\mathbf{o}, \Theta, M_{\mathbf{o}})}_{\text{outside the mask}}, \tag{8}$$

where $\mathbf{O}^{\text{in}} = \{\mathbf{o}_u\}$, *i.e.*, only using the user-chosen view. Although it is possible to consider the color guidance from more sampled views (*i.e.*, add more views in $\mathbf{O}^{\text{in}}$), it will introduce view inconsistency as will be discussed in our ablation study. $l_{\text{color}}(\mathbf{o}, \Theta, M)$ measures the color difference between the rendered image $F_{\Theta}^{\text{image}}(\mathbf{o})$ and the guiding color image $I_{\mathbf{o}}^{G}$ from view $\mathbf{o}$ with respect to the mask M:

$$l_{\text{color}}(\mathbf{o}, \Theta, M) = (F_{\Theta}^{\text{image}}(\mathbf{o}) - I_{\mathbf{o}}^{G})^2 \odot M. \tag{9}$$

The first part of Equation 8 measures the reconstructed quality inside the masks and the second part measures the reconstructed quality outside the masks.

Depth-guiding loss While we can obtain visually plausible inpainting results only using the color-guiding loss, it often generates incorrect depth, which might cause incorrect geometry and keep some unwanted regions in the scene. To fix these incorrect geometries, we introduce a depth-guiding loss, which is defined as:

$$L_{\text{depth}}(\Theta) = \sum_{\mathbf{o}\in\mathbf{O}} (F_{\Theta}^{\text{depth}}(\mathbf{o}) - D_{\mathbf{o}}^{G})^2, \tag{10}$$

where $D_{\mathbf{o}}^{G}$ is the guiding depth image from sampled view $\mathbf{o}$.

Experiments and evaluations

In this section, we show qualitative results on LLFF [12] and our own datasets, followed by ablation studies.

Implementation details

In this paper, we use the same architecture as the original NeRF [1] as the backend, and implement our framework in PyTorch and Python 3.9. Our framework is tested on a machine with an Intel Core i7-7800X CPU and an NVIDIA GeForce GTX-1080Ti GPU to train our models. For each scene, we first train a NeRF using random parameters and optimize it for 200,000 steps with a batch size of 4,096 using Adam [21], which takes about 18 to 20 hours. The sample points used in the fine and coarse models are 128 and 64, respectively. To inpaint each scene, we optimize Equation 7 for 50,000 steps which take about five hours.

Evaluation

Datasets To verify the performance of our framework, we create a customized dataset that contains three scenes: **figyua**, **desuku**, and **terebi**. The purpose is to obtain the ground truth of the NeRF inpainting task. For each customized scene, we collect a pair of photo sets, *i.e.*, (**original** and **removed**). For **original** set, we keep all objects in the scene and take photos from 24 camera positions. For **removed** set, we remove one object in the scene manually and take photos from another 24 camera positions. Although it is ideal for taking images from exactly the same camera positions, we found it very challenging to achieve in real-world scenes. To address this issue, we performed an additional alignment process using iNeRF [22] between the **original** and **removed** sets. Noted that although the concurrent NeRF inpainting works [10], [11] introduced datasets that serve the similar purpose, the datasets are not released publicly. Thus, we will use our own dataset to evaluate our method qualitatively and quantitatively.

Experiment setup As this is the first work focus on free-form inpainting on NeRF, we propose two baseline methods for comparisons:

baseline1: per-view color updating. We update the pretrained NeRF model F_{Θ} with all guiding images $\mathbf{I}^{G}$ by optimizing

$$\tilde{\Theta} \coloneqq \arg\min_{\Theta} \sum_{I_s^G \in \mathbf{I}^G} (F_{\Theta}^{\text{image}}(\mathbf{o}_s) - I_s^G)^2. \tag{11}$$

baseline2: masked mse retraining. We re-train a new NeRF using all guiding images $\mathbf{I}^{G}$ by optimizing:

$$\tilde{\Theta} \coloneqq \arg\min_{\Theta} \sum_{I_s^G \in \mathbf{I}^G} (F_{\Theta}^{\text{image}}(\mathbf{o}_s) - I_s^G)^2 \odot M_s. \tag{12}$$

Both *baseline1* and *baseline2* did not consider depth information during updating the pretrained NeRF model or re-train a new NeRF.

We compared the inpainting results of our framework to those of the two baseline methods on the LLFF dataset and our customized dataset. For the LLFF dataset, because there is no ground truth of the inpainted results, we perform qualitative evaluation by applying the three methods on each pretrained NeRF.

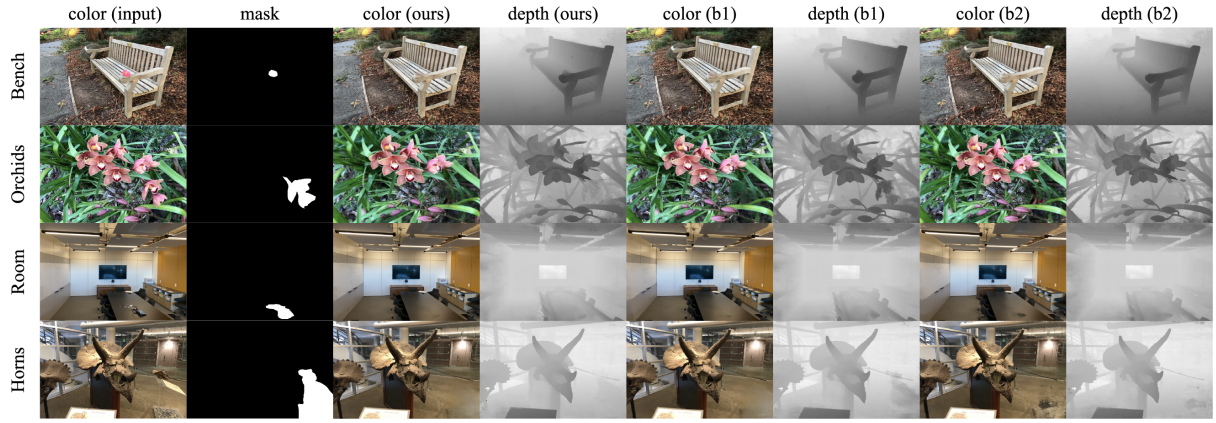

FIGURE 3: **Qualitative comparison - LLFF dataset.** For each scene, we show the user-chosen view image and the user-drawn mask on the left. We then show the color image and depth image generated by different methods: our method (ours), *baseline1* (b1), and *baseline2* (b2). The depth map of b1 still keeps the depth of the unwanted object. Meanwhile, the color of b2 might cause noise or shadow on the scene (shown in horns). Our method, compared to these two baselines, have better color and correct geometry on final results.

Meanwhile, we evaluate the multi-view consistency of the inpainted NeRF using the semantic consistency loss. More specifically, we randomly render 100 views of a NeRF (F_Θ) and compute the following average normalized distance as our multi-view consistency metric:

$$D_{mvc}(F_\Theta) = \frac{1}{N}\sum_{p,q}(\|\eta(I_p) - \eta(I_q)\|_2^2), \qquad (13)$$

where I_p, I_q is an image pair from two different views p, q and η is a Vision Transformer (ViT) feature extractor.

For our customized dataset, we also perform qualitative and the multi-view consistency evaluation. Moreover, we obtained the ground truth inpainted results using the following registration process; thus, we further perform quantitative evaluations. For a scene A in our customized dataset, we first trained a NeRF model F^A using the **original** dataset. For each image in the **removed** dataset, we obtained its camera position in the coordinate system of F^A. After performing our optimization, we compare the rendered results from the obtained camera position and its corresponding image in the **removed** dataset.

Results and discussions **LLFF dataset** We show the qualitative comparison between our method and the two baseline methods using the LLFF dataset in Figure 3 and Figure 5. In Figure 3, we observed that the depth maps of the inpainted NeRF generated by *baseline1* did not match the inpainted image content. In Figure 5, we showed that there are obvious visual inconsistencies between different views in the results generated by *baseline1*. To avoid these visual inconsistency, we choose to provide color guidance using only the user-chosen view and let the NeRF model maintain the view consistency by itself. *Baseline2* recovers visual satisfactory image content without any color guidance inside the masked region. However, **baseline2** still generates results that loss fine structures or synthesize some unnatural patches at complicated regions, which can be observed at Horns and Orchids. Our method achieved a multi-view consistency score of 0.143, outperforming baseline1 and baseline2 with scores of 0.343 and 0.355, respectively.

Customized dataset We show the qualitative comparison between our method and the two baseline methods using our customized dataset in Figure 1 and Figure 4. In Figure 4, we can observe that although *baseline1* can synthesize color content closer to the ground truth, it still fails to generate correct depth map. On the other hand, *baseline2* recovers the content in the masked region guided by the content from different views but still creates noisy and blurry result. Our framework generates closer color and depth images to the ground truth compared to the two baseline methods. Quantitatively, we evaluate the quality of the color image content inside the masked region of each inpainted color image using PSNR and LPIPS. For each metric, we compute the average over all images in the captured sequence of a scene, and then average the metric over all scenes. We also compute the L_1 distance between the inpainted and the ground truth depth images inside the mask. In Table 1, we showed that our proposed method is superior to the two baseline methods.

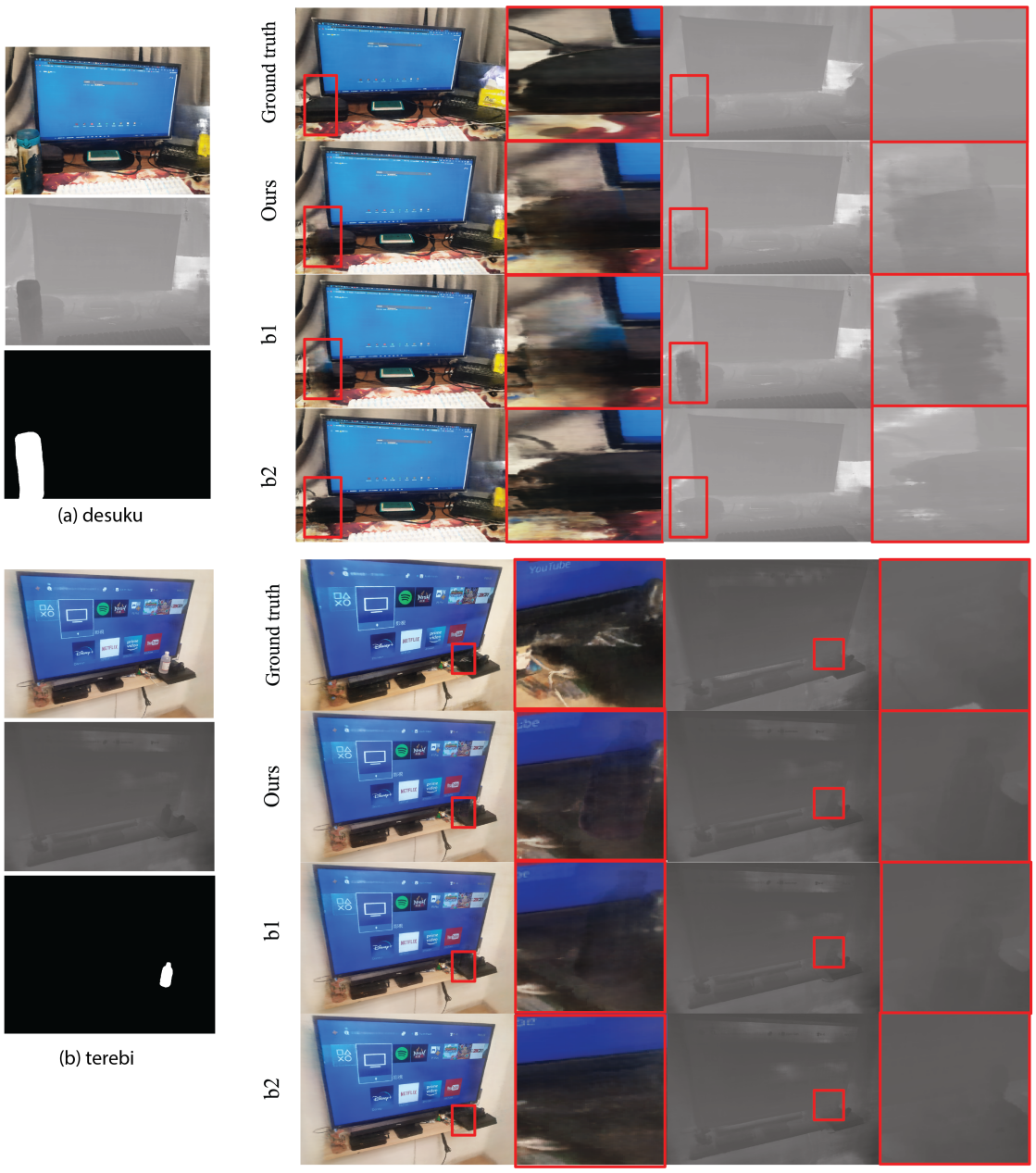

FIGURE 4: **Qualitative comparison - customized dataset.** For each customized scene, we demonstrate the ground truth, results generated by our framework, *baseline1* (b1), and *baseline2* (b2). Our framework generates more accurate depth maps and synthesizes more fine structures compared to *baseline1*. Compared to *baseline2*, our framework synthesizes more realistic and shape results.

FIGURE 5: **Qualitative comparison - visual consistency.** The rendered views generated by *baseline1* have severe visual inconsistency across different views (within the red box region). Meanwhile, our method synthesize visual consistent results across different views.

	PSNR↑	LPIPS↓	L1↓	MVC↓
Our	**28.42**	**0.081**	**0.165**	**0.113**
Baseline1	26.55	0.088	0.202	0.245
Baseline2	25.73	0.094	0.217	0.268

TABLE 1: We validate our framework by comparing to the baseline methods.

Ablation study

How important is the depth-guiding loss? Introducing the depth-guiding loss (L_{depth}) is one of the major contributions of our framework. We validate its effectiveness by comparing with the optimization results using color-guiding loss (L_{color}) only, depth-guiding loss (L_{depth}) only, and both losses.

The results are shown in Figure 6. We observed that optimizing using L_{depth} only already leads to correct geometries inside the masked region but introduces color noises outside. Our method optimizes both losses and generates correct geometries without color noises. In Figure 7, we can also observed that the unwanted object in the region with high depth variations can be removed by using L_{depth} only (red box). However, using L_{depth} only loses color information in the flat region (blue box). Our method combines these two losses and remove the unwanted object without losing color information in the flat region.

How important is color-guiding within the masked regions from sampled views? In our original color-guiding loss formulation in Equation 8, we only consider the color-guiding within the masked region on the user-chosen view (*i.e.*, $\mathbf{O}^{\text{in}} = \{\mathbf{o}_u\}$). Here, we validate this design choice by adjusting the number of views we consider the color-guidings within the masked region during the optimization.

We compared the results of following three settings:

1) only *user-chosen* view is used to guide the color inside the masked region, *i.e.*, $\mathbf{O}^{\text{in}} = \{\mathbf{o}_u\}$;
2) *three sampled views* are used to guide the color inside the masked region, *i.e.*, $\mathbf{O}^{\text{in}} = \{\mathbf{o}_i, \mathbf{o}_j, \mathbf{o}_k\}$ where i, j, k are randomly sampled;
3) *all sampled views* are used to guide the color inside the masked region, *i.e.*, $\mathbf{O}^{\text{in}} = \mathbf{O}$.

As shown in Figure 8, we observed that more visual inconsistency will be introduced when we use more inpainted images as color guidance. Our framework obtains stable results for most of the scene using user-chosen view only; thus, we choose to not to consider other inpainted regions during the computation of Equation 8.

Limitations and future work

More robust guidances and masks. Our framework generates initial guiding materials using existing methods, so also shares their limitations. For example, our framework fails to inpaint the image region with high reflectance content (Figure 9) or with a thin structure. It is possible to design a fusion method to fuse color and depth guidances from multiple methods to overcome individual limitations. Meanwhile, we used fixed masks and guiding materials during the optimization. This is sub-optimal when the unwanted object is occluded in some views. We plan to extend our framework to update the masks in every optimization step.

Volume feature for mask transferring. Our current framework uses the existing video-based object segmentation method to transfer the user-drawn mask. It is possible to perform mask transferring by conducting 3D volume segmentation using the volume feature extracted from the pretrained NeRF.

Conclusion

In this paper, we propose the first framework that enables users to remove unwanted objects or retouch undesired regions in a 3D scene represented by **a pretrained NeRF** without accessing its training images. Moreover, our framework requires no additional category-specific data and training. Instead, we formulated a novel optimization to inpaint a pretrained NeRF with the generated RGB-D guidances which is agnostic

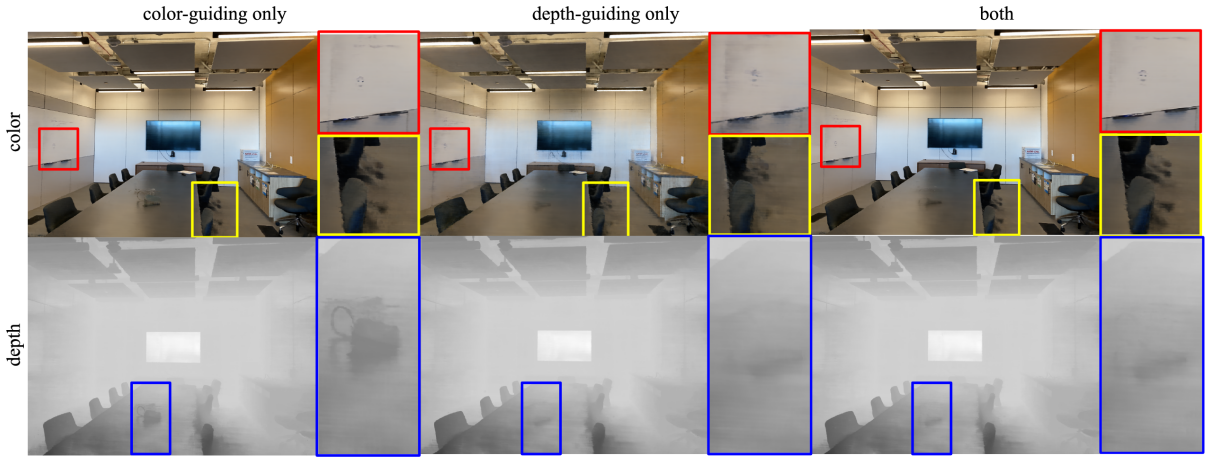

FIGURE 6: **Depth-guiding ablation result.** We show the optimization results using different guiding losses. We observed that the geometry of the unwanted object could not be removed using the color-guiding term only (depth inside the blue box). On the other hand, using the depth-guiding term only helps to get correct geometry but introduces color noises outside the masked region (red box and yellow box). Our method combines both terms to generate correct geometry (blue box) without introducing any color noises (red box and yellow box).

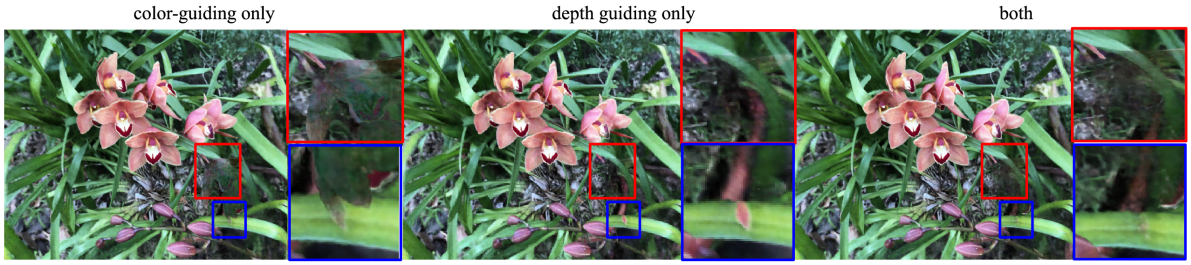

FIGURE 7: **Depth-guiding losses discussion.** The unwanted object in the region with high depth variations can be removed by using the depth-guiding loss (red box). However, using the depth-guiding loss only loses color information in the flat region (blue box). Combining both losses removes the unwanted object without losing any color information.

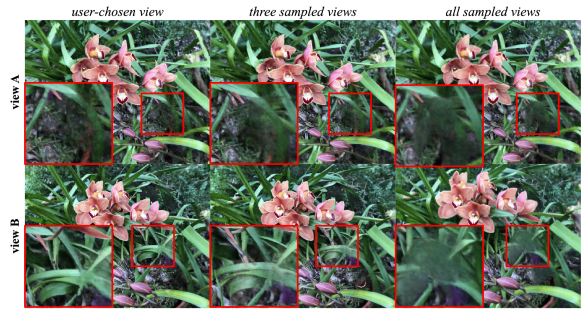

FIGURE 8: **Color-guiding ablation result.** The visual inconsistency becomes larger in the masked regions (red box) when the number of color guidances within the masked regions increased during optimization.

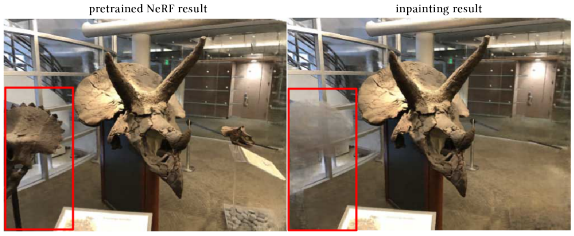

FIGURE 9: Our framework fails to inpaint the mask region in the left region (red box) and introduces artifacts in the optimized results.

to the NeRF architectures. We demonstrated that our framework handles a variety of scenes well, and also validate our framework using a customized dataset where the unpainted ground truth are available. We believe our framework takes the first step toward the pretrained NeRF editings and there are many frutiful editing applications to be explored.

Acknowledgement

This work was supported in part by JSPS Grant-in-Aid JP23K16921, JP21F20075, and National Science and Technology Council, under Grant, NSTC111-2634-F-002-022, 111-2221-E-002-145-MY3, 111-3111-E-002-002, and 112-2218-E-002-029, and National Taiwan University (NTU112L900902).

REFERENCES

1. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.
2. C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5741–5751.
3. K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.
4. C. Gao, A. Saraf, J. Kopf, and J.-B. Huang, "Dynamic view synthesis from dynamic monocular video," *arXiv preprint arXiv:2105.06468*, 2021.
5. S. Liu, X. Zhang, Z. Zhang, R. Zhang, J.-Y. Zhu, and B. Russell, "Editing conditional radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5773–5783.
6. K. Kania, K. M. Yi, M. Kowalski, T. Trzciński, and A. Tagliasacchi, "CoNeRF: Controllable Neural Radiance Fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.
7. H. K. Cheng, Y.-W. Tai, and C.-K. Tang, "Rethinking space-time networks with improved memory coverage for efficient video object segmentation," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
8. C. Cao and Y. Fu, "Learning a sketch tensor space for image inpainting of man-made scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 509–14 518.
9. J. T. Barron and B. Poole, "The fast bilateral solver," *ECCV*, 2016.
10. A. Mirzaei, T. Aumentado-Armstrong, K. G. Derpanis, J. Kelly, M. A. Brubaker, I. Gilitschenski, and A. Levinshtein, "Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields," *arXiv preprint arXiv:2211.12254*, 2022.
11. S. Weder, G. Garcia-Hernando, A. Monszpart, M. Pollefeys, G. Brostow, M. Firman, and S. Vicente, "Removing objects from neural radiance fields," 2023.
12. B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, 2019.
13. Z. Ren, A. Agarwala†, B. Russell†, A. G. Schwing†, and O. Wang†, "Neural volumetric object selection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, († alphabetic ordering).
14. B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, "Learning object-compositional neural radiance field for editable scene rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 13 779–13 788.
15. Q. Wu, X. Liu, Y. Chen, K. Li, C. Zheng, J. Cai, and J. Zheng, "Object-compositional neural implicit surfaces," in *European Conference on Computer Vision*. Springer, 2022, pp. 197–213.
16. D. Simakov, Y. Caspi, E. Shechtman, and M. Irani, "Summarizing visual data using bidirectional similarity," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
17. Y. Li, S. Liu, J. Yang, and M.-H. Yang, "Generative face completion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3911–3919.
18. S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–14, 2017.
19. G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 85–100.
20. R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 2149–2159.
21. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
22. L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "iNeRF: Inverting neural radiance fields for pose estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

I-Chao Shen is an assistant professor at the Graduate School of Information Science and Technology at the University of Tokyo, working with Takeo Igarashi. He did his Ph.D. in the computer graphics group at National Taiwan University, advised by Robin Bing-Yu Chen. He received B.B.A and M.B.A degrees in information management from National Taiwan University, in 2009 and 2011, respectively. He was a JSPS postdoctoral researcher, project assistant professor at The University of Tokyo. Please contact him at ichaoshen@g.ecc.u-tokyo.ac.jp .

Hao-Kang Liu is an software engineer in Synopsys, Inc, Taiwan. He was a master student in CMLab, National Taiwan University, Taiwan.

Bing-Yu Chen aka Robin Chen, received the B.S. and M.S. degrees in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree in Information Science from The University of Tokyo, Tokyo, Japan, in 2003. He is currently a Distinguished Professor with Department of Computer Science and Information Engineering, Department of Information Management, and Graduate Institute of Networking and Multimedia of National Taiwan University (NTU), and also the Dean of NTU College of Innovation and Design (D-School). His current research interests include Human-Computer Interaction, Computer Graphics, and Image Processing. He is a senior member of ACM and IEEE.