InNeRF360: Text-Guided 3D-Consistent Object Inpainting on 360° Neural Radiance Fields

Dongqing Wang Tong Zhang* Alaa Abboud Sabine Süsstrunk
School of Computer and Communication Sciences, EPFL
Lausanne, Switzerland

Abstract

We propose InNeRF360, an automatic system that accurately removes text-specified objects from 360° Neural Radiance Fields (NeRF). The challenge is to effectively remove objects while inpainting perceptually consistent content for the missing regions, which is particularly demanding for existing NeRF models due to their implicit volumetric representation. Moreover, unbounded scenes are more prone to floater artifacts in the inpainted region than frontal-facing scenes, as the change of object appearance and background across views is more sensitive to inaccurate segmentations and inconsistent inpainting. With a trained NeRF and a text description, our method efficiently removes specified objects and inpaints visually consistent content without artifacts. We apply depth-space warping to enforce consistency across multiview text-encoded segmentations, and then refine the inpainted NeRF model using perceptual priors and 3D diffusion-based geometric priors to ensure visual plausibility. Through extensive experiments in segmentation and inpainting on 360° and frontal-facing NeRFs, we show that our approach is effective and enhances NeRF's editability.

1. Introduction

Recreating and manipulating real-world scenarios is one of the main focuses of Virtual and Augmented Reality (VR/AR) applications. Neural Radiance Field (NeRF) and its variants [2, 23, 28] can efficiently model 360°real-world scenes for photorealistic novel view synthesis. Consequently, they have the potential to become widely accessible tools for representing the 3D world.

A desired feature of such applications is the ability to modify the content of the created scene, including object removal. However, direct inpainting in the NeRF framework is intractable due to the implicit representation of the captured scenes encoded through the weights of multilayer perceptrons (MLP), which hinders explicit user control of scene contents. For explicit NeRF variants, the scene representation of radiance fields contains ambiguous surface that is hard to segment with a bounding region.

Figure 1. Given a pre-trained NeRF and a text to remove specific objects (e.g.*"Remove the flowerpot and flowers"*), InNeRF360 produces accurate multiview object segmentations, and outputs an inpainted NeRF with visually consistent content.

Existing works on selecting [35] or removing [18, 26, 43] objects in a trained NeRF tackle the 3D problem by utilizing 2D input. These methods begin with sparse user inputs, then use 2D image/video segmentation [7, 10] for multiview segmentation and inpaint on RGB-D sequence. They are restricted to frontal-facing viewing angles, as the input scribbles or masks cannot extrapolate across different viewpoints in 360° scenes where the shape of the object

*Corresponding author.

can change drastically. Moreover, in the case of object occlusions on 360° scenes, inpainting 2D depth maps is inadequate for geometric supervision as it leads to inconsistencies in scene geometry.

In contrast, InNeRF360 is an inpainting pipeline with a depth-guided segmentation method dedicated to accurate object-level editing on 360° scenes. Instead of extrapolating sparse 2D input to 3D, InNeRF360 encodes text input into a promptable segmentation model, Segment Anything Model (SAM) [15], leveraging its accurate semantic segmentation. Our method is driven by the intuition that an object's semantic identity is *more consistent* over different viewpoints than its geometry. However, text-based 2D semantic segmentations may not always maintain consistency across views. To overcome this, we refine object masks using inverse depth space warping in 3D space across viewpoints, utilizing the consistent 3D positioning of objects.

Using view-consistent masks, we train a NeRF from scratch on the multiview inpainting from a 2D image inpainter [36]. The multiview training images slightly differ in the inpainted regions, accumulating into cloudy artifacts, i.e. floaters [42], in the new NeRF. To eliminate floaters, we finetune the scene guided by 3D diffusion priors trained on extensive geometric shapes [6] to determine whether density should be removed or incremented in a local voxel. The texture for the removed region is optimized by contextual appearance priors [45] from the surrounding regions of the segmentation. This creates a perceptually consistent 3D inpainted region that seamlessly blends into the scene. Extensive experiments show that InNeRF360 can effectively inpaint both 360° [2] and front-facing [22] real-world scenes, with potential to be extended into 3D editing tools.

To summarize, our contributions are as follows:

- InNeRF360 is the first work to achieve text-guided object inpainting in 360° NeRF scenes, ensuring visually consistent inpainted regions.
- Our approach efficiently generates multiview consistent 2D segmentation for 3D object inpainting through depth-warping refinement on initialized masks.
- We incorporate a 3D diffusion network as local geometric priors to remove artifacts in the inpainted region.

2. Related Works

Image Inpainting. Recent advancements in image inpainting focus on filling in masked regions to create visually consistent images. This primarily involves generative models like Generative Adversarial Networks (GANs) [13, 31, 46] and denoising diffusion models (DDPMs) [29, 36, 37]. These models generate photorealistic predictions for the missing pixels. However, when applied to inpainting multiview renderings of 3D scenes [25, 26], they often generate different inpainted regions for nearby views. This is due to their lack of 3D understanding, presenting a challenge for inpainting 2D observations of 3D scenes. InNeRF360 leverages image inpainting to address inpainting 360° scenes, ensuring view consistency across viewpoints.

Inpainting Neural Radiance Fields. Using neural radiance fields [23] to represent 3D scenes has achieved high-quality, photo-realistic novel view synthesis. NeRFacto [39] is an architecture designed to optimize NeRF's performance on real-world data. It incorporates various recent advances in NeRF, including hash encoding [28] and per-image appearance encoding [19], among others.

However, object removal presents a challenge in NeRF due to the implicit scene representation by the underlying neural networks. Previous works [24, 40] utilize the supervision of Contrastive Text-Image Pre-Training (CLIP) [34]. They focus on inpainting a single object and cannot generalize to real-world scenes. Methods utilizing depth-based approaches [18, 25, 26, 43] remove objects with user-drawn masks and depth sequences, and inpaint missing regions naively with 2D image inpainter on the training images of NeRF. These approaches are limited to front-facing scenes for two reasons. First, their segmentation relies on the quality of initialize masks by supervised video object segmentation methods [4, 5, 7, 30], which struggle on temporal consistency across frames for challenging cases such as transparent objects. These methods struggle to output accurate object masks in 360° scenes, where object shapes change drastically across views. Secondly, the chosen 2D inpainting methods output different inpainting across views. Such inconsistency results in floaters in the trained NeRF, and is much more pronounced on 360° scenes than on frontal-facing ones. In contrast, InNeRF360 enhances the consistency of multiview segmentation by utilizing semantical identity and object 3D location consistency, thereby producing accurate masks for desired objects. Moreover, InNeRF360 is designed for 360° NeRF inpainting by removing floaters from the inpainted NeRF with geometric priors, and inpaint with contextual perception guidance.

Text-Guided 3D Editing. Given the popularity of text-conditioned image generative models, many works focus on generating 3D content with text instructions. Some rely on joint embeddings of CLIP to synthesize 3D meshes [21, 27] or neural radiance fields [14, 16]. Others distill a pre-trained diffusion model to optimize NeRF scenes in the latent space [20, 33]. These methods all suffer from having to map the inconsistent 2D diffusion model outputs to a 3D-consistent scene. Instruct-NeRF2NeRF [9] edits renderings of a pre-trained NeRF model to preserve 3D consistency. However, it cannot remove scene objects or perform object-level editing as it operates in latent space for image editing. Our InNeRF360 operates in image space to accurately pinpoint and crop objects and allows removing an arbitrary number of objects from the scene through text instructions, while using 3D diffusion priors for local geometry finetun-

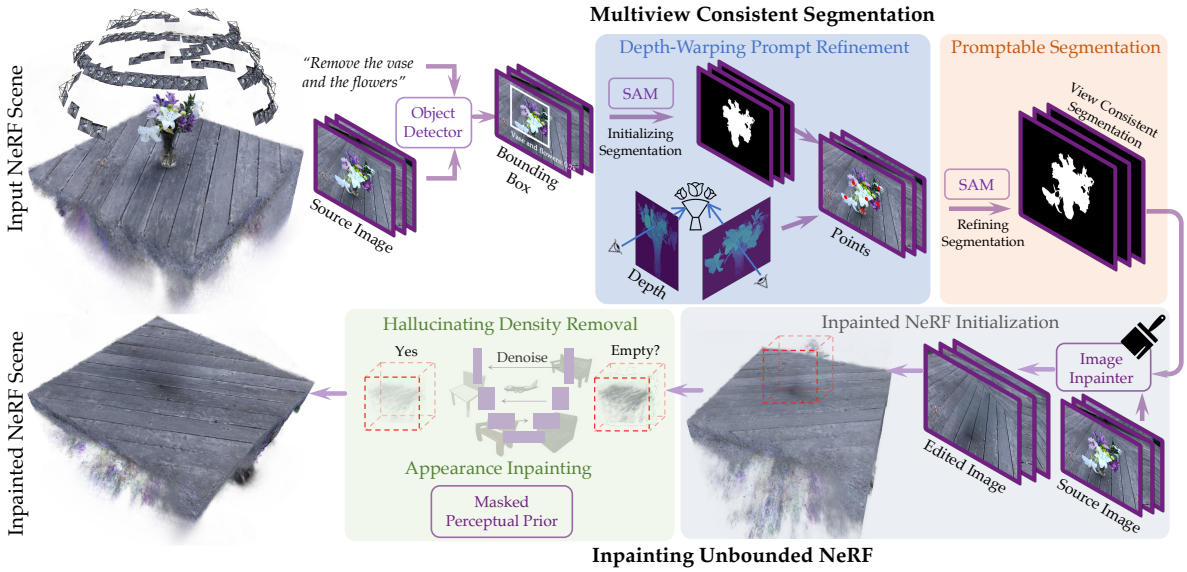


Figure 2. **Overview of InNeRF360 framework.** 1. *Multiview Consistent Segmentation.* We initialize masks using bounding boxes from the object detector, which encodes both the source image and text. With rendered depth from the input NeRF, we apply depth-warping prompt refinement to iteratively update points for the Segment Anything Model (SAM) to output view-consistent 2D segmentations. 2. *Inpainting 360° NeRF.* We obtain edited images through image inpainter with the masks and source images to retrain the inpainted NeRF. We then finetune the new NeRF model using a geometric prior trained from a 3D diffusion model and a masked perceptual prior.

ing to avoid the global inconsistency prior works exhibit.

3. Method

InNeRF360 takes as input a trained NeRF with source images which the model trained on, and an instructive text. It outputs the inpainted 3D scene with the desired object(s) removed and filled with a visually consistent background without artifacts. Our pipeline is shown in Fig. 2.

3.1. Background: Neural Radiance Fields

A Neural Radiance Field (NeRF) encodes a 3D scene as a function f_θ parametrized by an MLP with learnable parameters θ, which maps a 3D viewing position $\mathbf{x}$ and its 2D direction $\mathbf{d}$ to the density σ and a viewing-dependent color $\mathbf{c}$: $f_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$. Rendering a NeRF from a posed camera is done by sampling batches of rays for the camera pose, and rendering corresponding pixel colors for each ray. For each ray $\mathbf{r} = (\mathbf{o}, \mathbf{d})$, we sample an array of 3D points $(\mathbf{x}_i, t_i)$, $i = 1, 2, \cdots, K$, where $\mathbf{x}_i \in \mathbb{R}^2$ and t_i is the depth. We query the MLP with these points along the ray for $\{\sigma_i\}_{i=1}^K$ and $\{\mathbf{c}_i\}_{i=1}^K$.

The estimated RGB of ray $\hat{\mathbf{C}}(\mathbf{r})$ is obtained by alpha compositing [23] the densities and colors along the ray:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{K} \alpha_i T_i \mathbf{c}_i, \tag{1}$$

where $T_i = 1 - \exp(-\sigma_i \|t_i - t_{i-1}\|)$ is the ray transmittance between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, and $\alpha_i = \prod_{j=1}^{i-1} T_i$ is the attenuation from ray origin to $\mathbf{x}_i$. The MLP is optimized through pixel loss for the distance between the estimated pixel value and the ground truth color.

3.2. Multiview Consistent Segmentation

The first stage is to obtain refined multiview segmentation masks for the objects to remove/edit given by the text input. We take as input a pre-trained NeRF model and N source images given by the set of N source camera poses.

We initialize segmentations through the Segment-Anything Model (SAM) [15] with the bounding boxes given by Grounded Language-Image Pre-training (GLIP) from a large-scale dataset of image-text pairs [17]. GLIP: $\mathcal{D}(\mathbf{I}, \mathbf{s}) = \{(B_q, p_q)\}_{q=1}^Q$ takes in an RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ and a text $\mathbf{s}$, and encodes each through respective image and language encoders. Q is the number of bounding boxes. We then take these bounding boxes $\{B_q\}_{q=1}^Q$ from the model output in which $B_q = (l_q, r_q, u_q, d_q) \in \mathbb{R}^4$ with corresponding probability p_q. These boxes are sometimes inaccurate and fail to enclose the desired region, see Fig. 3 (b). Hence, we propose our depth-based prompt refinement for consistent segmentation.

Depth-Warping Prompt Refinement. "Depth Warp" operates on the sampled points to enhance segmentation accuracy. It leverages the depth information inherited from the input NeRF and projects these points back into the pixel space to align them with other 2D observations within the scene. *We thereby establish a cohesive depth constraint across different views.* The depth for a sampled ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ with origin $\mathbf{o}$ and direction $\mathbf{d}$ in a trained NeRF

scene can be estimated via a modification of Eq. (1):

$$D(\mathbf{r}) = \sum_{i=1}^{I} \alpha_i T_i \cdot t_i, \tag{2}$$

For each training view, we randomly select m other training views. From each selected view, we sample a fixed number p of rays that correspond to pixels within the mask region. Given a ray $\mathbf{r}$ and its estimated depth from Eq. (2), we compute the 3D location of the selected point prompt. This information is then mapped back to the current training view. Rays and their corresponding point prompts whose depth is above a certain threshold are discarded, as they may represent background objects misclassified as foreground sections to be removed. If the corresponding pixel on the current view falls outside the masked region in the current view, we add this point prompt to the current view. Subsequently, we pass this view with the new sets of point prompts to SAM for refined segmentation.

By generating these "out-of-the-box" point-based prompts for the segmentation model with our NeRF-based depth prior, we ensure that the object is accurately segmented even in cases where the initial bounding box information is insufficient or incomplete from certain viewpoints. This approach greatly increases the segmentation masks' accuracy for desired objects in the scene, see Fig. 3.

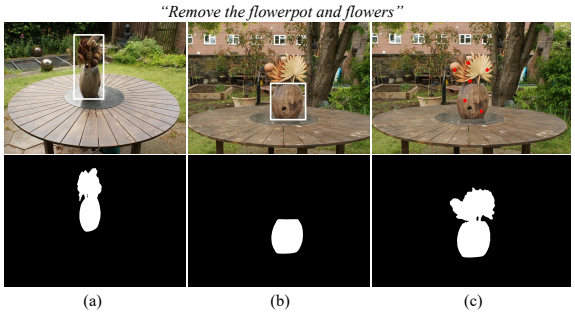


Figure 3. **Inconsistent bounding box across different views.** (a) and (b) are from the same dataset under the same instruction. However, the generated bounding boxes are different. After applying depth warping refinement (point prompts as red dots), (c) generates accurate segmentation.

Promptable Segmentation After obtaining the refined point-based prompts, we employ SAM to identify and segment all the rendered observations $\{\mathbf{I}_n\}_{n=1}^{N}$ for refined masks $\{\mathbf{M}_n\}_{n=1}^{N}$. Specifically, SAM takes as input an image $\mathbf{I}$ and a prompt in the form of j points pinpointing the object o_q, and produces an accurate segmentation mask $\mathbf{M_q}$ of the same size as $\mathbf{I}$: $\mathbf{M}_q = \texttt{SAM}(\mathbf{I}, o_q)$. For each image $\mathbf{I}_i \in \{\mathbf{I}_n\}_{n=1}^{N}$, we get a union binary mask for all the objects to be removed:

$$\mathbf{M}_i = \bigcup_{q=1}^{Q} \mathbf{M}_q. \tag{3}$$

3.3. Inpainting 360° NeRF

Inpainted NeRF Initialization. With the rendered observations $\{\mathbf{I}_n\}_{n=1}^{N}$ and corresponding masks $\{\mathbf{M}_n\}_{n=1}^{N}$, we adopt a 2D image inpainter [36] to edit each observation as priors for the optimization. We then initialize the inpainted scene with Nerfacto [39]. This architecture is designed to optimize the performance of NeRF on real-world image captures. However, each inpainted image has varying pixel-level content despite being perceptually plausible as a standalone image. Therefore, relying solely on RGB supervision leads to floaters in the NeRF (Fig. 4). To produce clean and perceptually consistent inpainting, we finetune the initialized NeRF with both geometry and appearance priors.

Figure 4. **Examples of artifacts in the initialized NeRF.** As the 2D inpaintings contain different inpainted pixels, their inconsistencies accumulate in the 3D inpainted region as floater artifacts.

Hallucinating Density Removal. The density artifacts produced through inconsistent 2D inpainting can be seen as floaters. We train a denoising diffusion probabilistic model (DDPM) [12] on ShapeNet [6] to iteratively denoise a m^3 resolution voxel grid of discretized binary occupancy x as a local 3D geometry prior: given a NeRF density σ at timestep t, $x_t = 1$ if $\sigma > \rho$ else $x_t = -1$, where ρ is a chosen threshold for whether a voxel is empty. For training, from each ShapeNet mesh we randomly select N cubes that encompass 3% to 8% of the object bounding volume, and voxelize them to m^3 resolution. The loss function for the diffusion model is given by the MSE loss between the true noise ϵ and the predicted noise ϵ_θ where θ parameterizes the diffusion model U-Net:

$$\mathcal{L}_{\text{ddpm}}(\theta) = \mathbb{E}_{t,x_0,\sigma}[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2], \tag{4}$$

where $t \in [1, 1000]$ is the number of timesteps in the noising diffusion process, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ where α_t is a function of the timestep t that parameterized transitions from x_0 to x_{1000} (i.e. a noise schedule).

The Density Score Distillation Sampling (DSDS) [42] loss is defined to penalize regions with density $\sigma > \rho$ that the trained diffusion model deems as empty, and regions that are empty where $x_t = 1$ predicted by the model:

$$\mathcal{L}_{\text{DSDS}} = \sum_i u_i \sigma_i + (1 - u_i) \max(w - \sigma_i, 0), \tag{5}$$

where $u = \mathbb{1}\{x_0 < 0\}$, and w is a chosen upper limit for density σ in occupied voxels.

The original sampling of the DSDS loss is by sampling a low-resolution density grid stored through ray bundles. During training, the grid selects the center of 3D cubes to be voxelized and enter the diffusion process. The grid gets updated by a visibility field determining what it sees in the NeRF scene. Our aim is to focus on removing floaters within the inpainted region. Therefore, we apply the refined segmentation mask from the image space to limit the visibility field to only look at the inpainted regions, and eliminate sampled rays corresponding to pixels outside the mask:

$$\mathcal{L}_{\text{geom}} = \sum_j (u_j \sigma_j + (1 - u_j) \max(w - \sigma_j, 0)) \cdot V_j, \quad (6)$$

where V is an indicator function whose value is 1 for each voxel cube whose center is located within the current "visible region" given by the segmentation masks. Consequently, we sample cubes near the inpainted region. The geometric prior trained on extensive shapes in [6] preserves the original surface (e.g., the table supporting the removed flowerpot) while penalizing the floaters in the inpainted region created through inconsistent 2D inpainting.

Perceptually-Consistent Appearance Inpainting. The above method removes floaters created by inconsistent 2D inpainting, however, it does not produce visually consistent textures to fill in the removed region. Therefore, we utilize a patch-based loss [45] to enhance the model's robustness and alleviate blurring effects.

Specifically, we sample all the pixels from the input inpainted images to get W image patches $\{P_w\}_{w=1}^{W}$ with the size of v^2. These patches, $\{P_w\}_{w=1}^{W}$, can be divided into two non-overlapping groups of patches P_{w_i} and P_{w_o} depending on whether a patch *contains* pixels within the inpainted region. For patches in P_{w_o}, we apply pixel-wise L1. This pixel loss is obtained by comparing the RGB values of each pixel p in the inpainted patch, denoted as $\tilde{\mathbf{C}}_p$, with the corresponding rendered RGB value, denoted as $\hat{\mathbf{C}}_p$,

$$\mathcal{L}_{\text{pix}} = \frac{1}{v^2 |P_{w_o}|} \sum_{p_\mathbf{r} \in P_{w_o}} \left\| \hat{\mathbf{C}}_p - \tilde{\mathbf{C}}_p \right\|_1. \quad (7)$$

For patches in P_{w_i} containing the inpainted region, we compute the perceptual similarity using LPIPS between the inpainted image patch $\tilde{P}_I$ and the corresponding patch P on the rendered image. We denote $\mathbf{C}_P$ as the set of pixel values in patch P, and define $\mathcal{L}_{\text{in}}$ as the inpainting loss.

$$\mathcal{L}_{\text{in}} = \frac{1}{|P_{w_i}|} \sum_{P \in P_{w_i}} \text{LPIPS}(\mathbf{C}_P, \mathbf{C}_{\tilde{P}_I}). \quad (8)$$

The inpainting loss measures the perceptual difference between the inpainted patch and the target inpainted image, while the pixel loss quantifies the pixel-level discrepancy between the inpainted and rendered RGB values. Together, these losses provide a comprehensive assessment of the reconstruction quality, accounting for both perceptual similarity and pixel-wise accuracy.

Loss Functions. Our optimization is the weighted sum of the geometric prior $\mathcal{L}_{\text{geom}}$ (Eq. (6)), pixel loss $\mathcal{L}_{\text{pix}}$ (Eq. (7)), appearance prior $\mathcal{L}_{\text{in}}$ (Eq. (8)) with $\lambda_{(\cdot)}$ as weight terms:

$$\mathcal{L} = \lambda_{\text{geom}} \cdot \mathcal{L}_{\text{geom}} + \lambda_{\text{in}} \cdot \mathcal{L}_{\text{in}} + \mathcal{L}_{\text{pix}}. \quad (9)$$

4. Experiments

In this section, we evaluate InNeRF360 on various real-world captured datasets for text-guided inpainting.

Datasets. We take 360-degree datasets from MipNeRF, MipNeRF-360, and NeRFStudio [1, 2, 39]. In addition, due to the absence of ground truth data for 360° scene inpainting, we capture new datasets with and without the object removed for *quantitative* evaluation. Segmentation mask ground truth does not exist for the 360° scenes we evaluate. We also compare with front-facing datasets from SPIn-NeRF [26] and IBRNet [41] to show that our method produces better inpainting over baseline methods. Details of the datasets we used are provided in Tab. 5.

Baseline. We select baselines based on specific tasks. SPIn-NeRF **SPN** [26] is the closest work to ours. For segmentation, we compare with the multiview segmentation from **SPN** and a recent video segmentation method **Dino** [5]. For the inpainting task, we compare our inpainting quality with our implemented version of SPN that works with 360-degree scenes **SPN-360**. We also compare with **per-frame image inpainting** [36] to show that our method generates more consistent inpainting across different viewpoints.

4.1. Segmentation Comparison

We qualitatively compare our segmentation results to SPN and Dino. As input, we give the first frame segmentation to both SPN and Dino, and give the corresponding text instructions for InNeRF360. As we are evaluating the extrapolation ability of each segmentation method over drastically different viewpoints, we select the 82nd frame from *vasedeck* and the 81st image from *room*. They represent the respective frames of the videos created from the dataset.

As shown in Fig. 5, Dino produces incomplete segmentation for unseen views in challenging scenes, such as the *vasedeck* featuring a transparent vase. In such cases, SPN struggles to generate complete segmentation when initialized with Dino's output. While we also initialize with a 2D segmentation model (SAM), it facilitates flexible, promptable input. This can be utilized by our depth-warping refinement to output point prompts specifically in the vase section of the image. These prompts guide SAM towards accurate segmentation, as elaborated in Sec. 3.2. For *room*, where only a part of the slippers is present in the image, Dino and SPN once again yield incomplete segmentations.

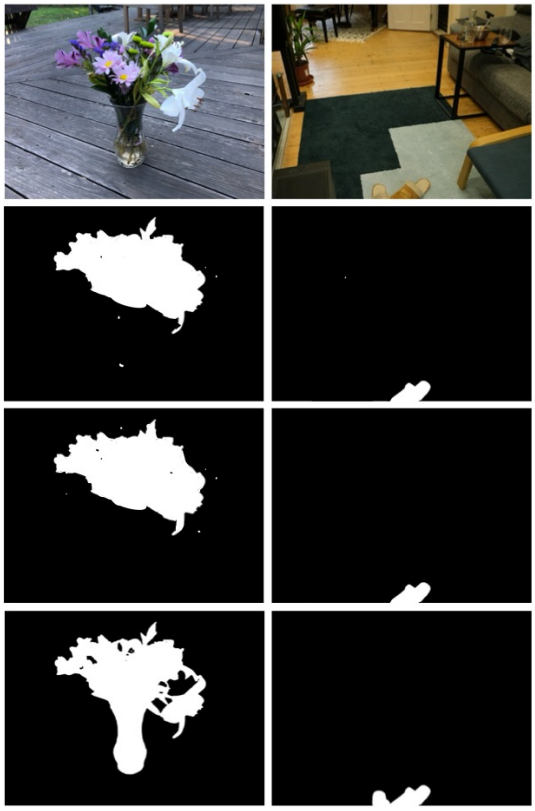


Figure 5. **Qualitative comparison on 3D object segmentation.** InNeRF360 outputs accurate masks for complex cases containing transparent (vase) or incomplete objects (partial slippers).

InNeRF360 is able to output complete object segmentation.

4.2. Inpainting Evaluation

Results on 360-degree scenes. InNeRF360 can handle a wide range of scenarios for object inpainting in 3D scenes, as depicted in Fig. 7. We encourage the readers to view supplementary videos to inspect the quality of our results.

InNeRF360 can remove *large scene objects against complex backgrounds*. In *Bear*, the background of the bear contains varying trees and branches, and the 2D inpainted images are therefore very noisy, as shown in Fig. 16. However, with our floater removal and appearance prior, InNeRF360 produces a clean stone surface in the final edited NeRF.

Occlusion and geometry deformation across varying viewpoints are major causes of 3D inconsistency in inpainting, and the datasets we use contain both scenarios. In *Room*, the window, wall, and the piano behind the score can lead to inconsistency. However, we generate view-consistent content to seamlessly fill the missing region.

Moreover, InNeRF360 is capable of inpainting *multiple objects* located *anywhere in the 3D scenes* without introducing blurry artifacts in the resulting NeRF scene. When given a text input containing multiple objects (*Room*: "slippers" and "piano sheet"; *Bulldozer*: multiple "cones"), InNeRF360 produces inpainted regions that seamlessly blend with the surrounding context, yielding visually coherent and high-quality inpainting results.

Figure 6. **Qualitative comparison with SPN-360.** Text: *Remove the vase and the flowers.* InNeRF360 inpaints clean and visually plausible regions while better preserving surrounding scenes.

	Cup		Starbucks	
Methods	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓
Per-Frame	0.6149	201.70	0.5981	260.93
SPN-360	0.6421	252.34	0.6278	215.28
NeRFacto	0.7328	271.56	0.6832	258.39
$+\mathcal{L}_{\text{in}}$	0.7137	210.57	0.6658	223.82
$+\mathcal{L}_{\text{geom}}$	0.6197	189.57	0.5795	166.45
$+\mathcal{L}_{\text{in}} + \mathcal{L}_{\text{geom}}$ (**Ours**)	**0.5377**	**159.76**	**0.4523**	**153.46**

Table 1. **Quantitative evaluation on the inpainting quality**. Our method achieves better results than baseline methods and our ablated settings on captured datasets.

Fig. 6 shows qualitative comparison to **SPN-360**. Our method not only synthesizes a perceptually-consistent inpainted region, but also preserves the surrounding background closer to the input NeRF scene. We speculate the reason for the background-preserving inpainting to be that SPN inpaints on depth maps with segmentation masks generated from RGB images. We elaborate on our choice not to use 2D depth map inpainting in the supplementary.

Ablation studies on our design choices. Our depth-warping method produces refined point-based prompts for the segmentation model, and outputs complete and consistent multi-view segmentation, as shown in Fig. 3.

Fig. 8 qualitatively ablates our choice of loss functions. In Fig. 8a (ii), the vanilla NeRFacto model outputs a concentrated artifact in the inpainted region along with noisy texture in nearby regions which we suspect is due to per-image appearance encoding on inconsistent 2D inpainted images. Fig. 8a (iii) shows NeRFacto $+\mathcal{L}_{\text{in}}$ which improves inpainted texture, but cannot reduce the floater artifact. These artifacts have view-dependent appearances from individual views and are therefore difficult to remove from appearance priors. In Fig. 8a (iv) for InNeRF360, we can see a clean and perceptually consistent surface in the edited

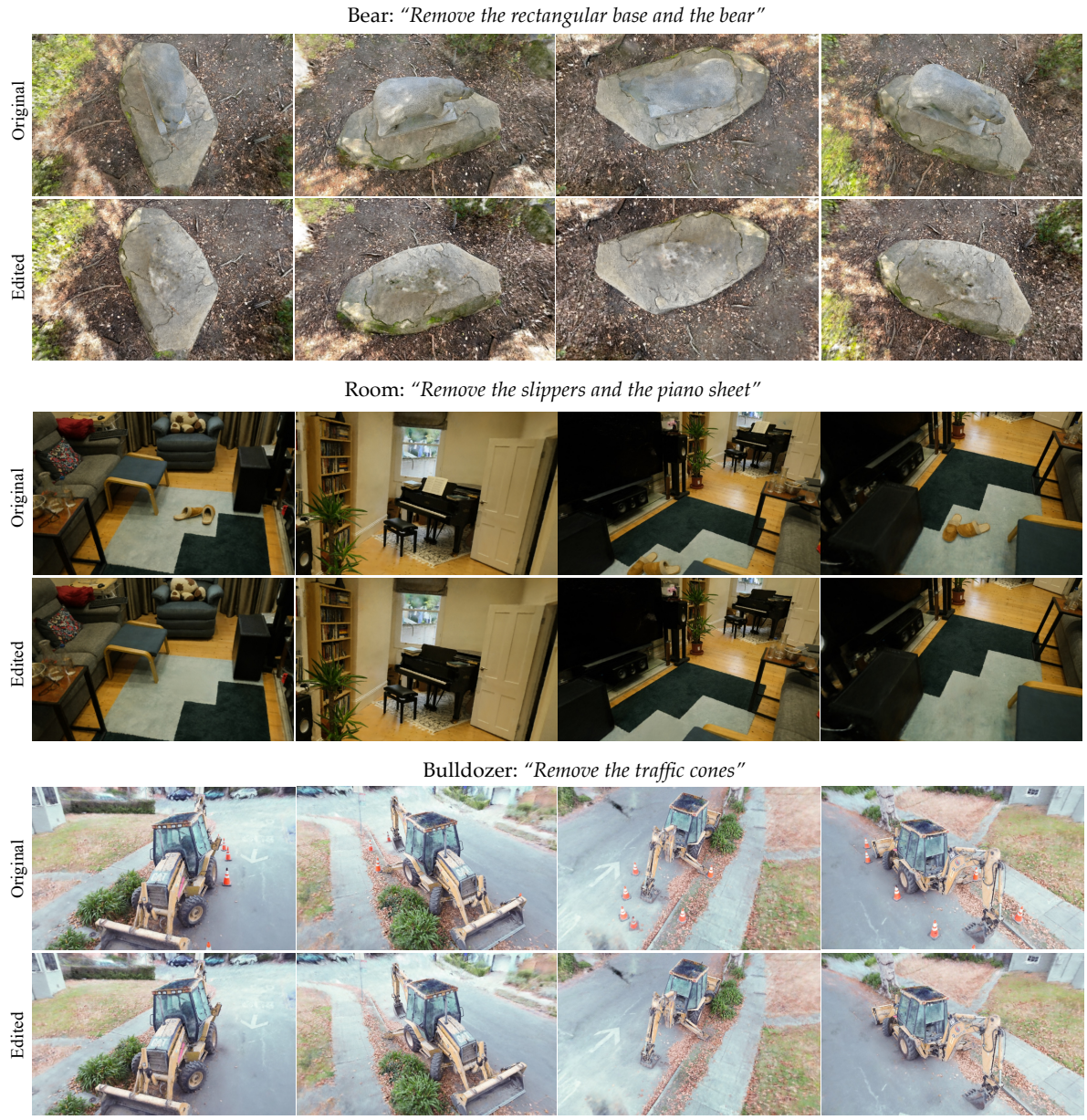


Figure 7. **Qualitative inpainting results on 360 scenes.** Our method works with various types of NeRF scenes. We can also remove arbitrary numbers of objects given the text input, independent of the complexity of the scene content.

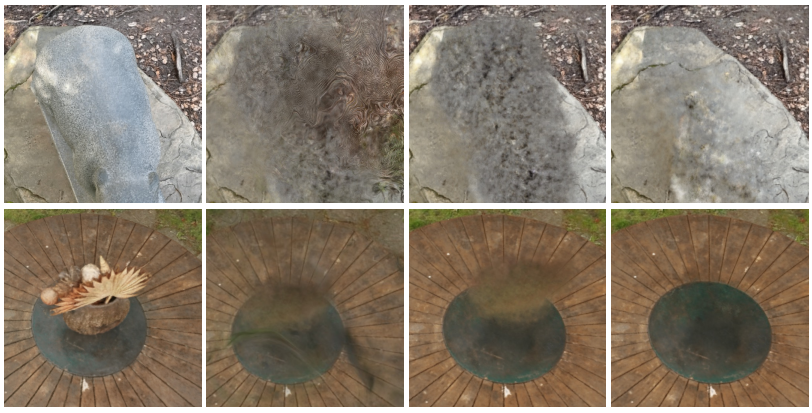
scene. In Fig. 8b shows InNeRF360 versus trained without $\mathcal{L}_{\text{in}}$. We can see that the LPIPS loss can improve blurry background output due to inconsistent 2D inpainting. The lower part of Tab. 1 shows quantitative ablation on each loss term in InNeRF360. Our complete architecture performs better than without each of the loss terms.

Inpainting quality. Due to the lack of baseline and ground truth datasets on inpainting 360° NeRF scenes, we captured real-world datasets for quantitative comparison on the quality of inpainted renderings. Since InNeRF360 generates consistent and complete 3D segmentation over baseline methods, the 2D inpainting initialization is naturally much less noisy than baseline methods.

Datasets	Garden	Room	Vasedeck	Bulldozer	Bear
Ours	**89%**	**71%**	**81%**	**83%**	**92%**
Per-frame	11%	29%	19%	17%	8%

Table 2. **User study comparing with per-frame inpainting on visual consistency between consecutive frames.** In each of the scenes, our inpainted NeRF renders higher view consistency than per-frame inpainting. Per-frame editing lacks a 3D understanding of each scene and inpaints each image independently.

We evaluate our inpainting quality on each frame of the



(i) Input Scene (ii) NeRFacto (iii) NeRFacto + $\mathcal{L}_{\text{in}}$ (iv) **Ours**

(a) Qualitative ablation for $\mathcal{L}_{\text{geom}}$ on *Bear* and *Garden*.

(I) NeRFacto + $\mathcal{L}_{\text{geom}}$ (II) **Ours**

(b) Qualitative ablation for $\mathcal{L}_{\text{in}}$ on *Garden*.

Figure 8. **Ablation for losses on geometric and appearance priors**. The artifact in the inpainted region is not as pronounced if viewed from aside as when viewed from the top. Our method is able to optimize an inpainted NeRF without artifacts and with a consistent and unblurry background.

Figure 9. **Editing comparison with In2n.** InNeRF360, combined with appropriate mask-conditioned image editing models, can generate accurate editing on desired objects. In contrast, In2n gives the wrong texture to unwanted regions.

renderings with per-frame inpainting and SPN-360. We report LPIPS [44] and Frechet Inception Distance (FID) [11] metric in Tab. 1 by comparing with the output of the captured empty scene rendered under the same camera trajectory which we use as ground truth. Our method outperforms each baseline method and outputs visually consistent inpainting without visual artifacts.

A quantitative baseline comparison to SPN on frontal scenes is also in the supplementary materials.

3D consistency over per-frame inpainting. A naive approach for 3D scene inpainting is to independently inpaint every rendered image of the scene with a 2D image inpainter. In contrast, InNeRF360 produces inpaintings with higher view consistency across all viewpoints.

We verify such a claim with a user study where participants were presented with two video clips of each inpainted scene, rendered with sequential camera trajectories. They were then asked to identify which clip appeared more consistent. Additional details about the user study can be found in the supplementary material. The results, presented in Tab. 2, clearly show that our rendered inpaintings exhibit superior temporal consistency compared to per-frame edits.

4.3. Editing Accuracy

As shown by Fig. 9, our segmentation module can be connected with a mask-conditioned image editor [8] to generate view-consistent editing with object-level control through text instructions, which InstructNeRF2NeRF (In2n) [9] cannot. However, note that editing is not the focus of our work. We show this result simply to demonstrate a possible extension to our method. Details are provided in the supplementary.

5. Limitations and Conclusion

Limitations. Our method inherits certain constraints of vision-language models. In scenarios where the text instruction cannot be accurately localized within the image, InNeRF360 may struggle in generating segmentation consistently aligned with the views. This issue arises when the initial masks provided by the 2D object detector are inaccurate or too noisy for effective refinement. Addressing this challenge is a focus of our future work.

Conclusion. In conclusion, we have presented InNeRF360, a unified system to accurately segment and inpaint objects in 360° NeRFs with text instructions. We synthesize perceptually consistent inpainting without artifacts and can extend to object-level stylization, improving the controllability of NeRF.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021. 5

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 1, 2, 5

[3] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402, 2023. 3

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, 2021. 2

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2, 5

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 4, 5

[7] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[8] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. 8, 3, 5

[9] Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19740–19750, 2023. 2, 8, 3

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017. 1

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 8, 1

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6840–6851. Curran Associates, Inc., 2020. 4

[13] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG) (Proceedings of SIGGRAPH)*, 36(4):1–14, 2017. 2

[14] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 867–876, 2022. 2

[15] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, 2023. 2, 3

[16] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 2

[17] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10965–10975, 2022. 3

[18] Hao-Kang Liu, I Shen, Bing-Yu Chen, et al. Nerf-in: Free-form nerf inpainting with rgb-d priors. *arXiv preprint arXiv:2206.04901*, 2022. 1, 2

[19] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7219, 2021. 2

[20] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *arXiv preprint arXiv:2211.07600*, 2022. 2

[21] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13492–13502, 2022. 2

[22] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG) (Proceedings of SIGGRAPH)*, 38 (4):1–14, 2019. 2

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3

[24] Ashkan Mirzaei, Yash Kant, Jonathan Kelly, and Igor Gilitschenski. Laterf: Label and text driven object radiance

fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–36, 2022. 2

[25] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A. Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G. Derpanis, and Igor Gilitschenski. Reference-guided controllable inpainting of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2, 6

[26] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G. Derpanis, Jonathan Kelly, Marcus A. Brubaker, Igor Gilitschenski, and Alex Levinshtein. SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 5, 3, 6, 7

[27] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *ACM SIGGRAPH Asia Conference Proceedings*, pages 1–8, 2022. 2

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG) (Proceedings of SIGGRAPH)*, 41(4):1–15, 2022. 1, 2

[29] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 2

[30] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 2

[31] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016. 2

[32] Polycam.Inc. YOLO by Ultralytics, 2020. 2

[33] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations (ICLR)*, 2023. 2

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 2

[35] Zhongzheng Ren, Aseem Agarwala†, Bryan Russell†, Alexander G. Schwing†, and Oliver Wang†. Neural volumetric object selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. († alphabetic ordering). 1

[36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 2, 4, 5, 7

[37] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH Conference Proceedings*, pages 1–10, 2022. 2

[38] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2149–2159, 2022. 7

[39] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH Conference Proceedings*, 2023. 2, 4, 5, 7

[40] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3835–3844, 2022. 2

[41] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5

[42] Frederik Warburg*, Ethan Weber*, Matthew Tancik, Aleksander Hołyński, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2, 4, 7

[43] Silvan Weder, Guillermo Garcia-Hernando, Áron Monszpart, Marc Pollefeys, Gabriel J. Brostow, Michael Firman, and Sara Vicente. Removing objects from neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16528–16538, 2023. 1, 2, 3, 6

[44] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8, 1

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 2, 5

[46] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. *arXiv preprint arXiv:2103.10428*, 2021. 2

InNeRF360: Text-Guided 3D-Consistent Object Inpainting on 360° Neural Radiance Fields

Supplementary Material

A. Additional Results

360-degree scenes. We encourage our readers to view the supplementary video for a comprehensive set of results from our evaluated datasets and to qualitatively assess our method. To quantitatively evaluate the output of InNeRF360 against ground truth inpainted scenes, we evaluate our method on captured real-world scenes. Fig. 10 demonstrates that even with occlusion between the plant and the object to remove, our method successfully generates clean inpainting with the rest of the scene unmodified.

To clarify our evaluation, we conduct quantitative experiments on the baselines *Ideal*, *ObjectNeRF-M*, *SPN-360-M* on our captured real-world datasets:*Starbucks*, *Glass Cat* and *4-Objects* The *ObjectNeRF-M* baseline involves training a NeRF with L2 loss outside the segmentation masks only. ObjectNeRF-M produces lower-quality inpainting results than our InNeRF360, evidencing that inpainting on 360-degree NeRF is more complicated than frontal-facing scenes. The implementation of SPN-360 is based on NeRFacto. Its segmentation masks and inpainting results are both generated by the method used in SPIn-NeRF so that we can compare them with our InNeRF360 results. This lets us contrast the performance of the complete methods. To address the concern for reliability, we provide evaluations with *SPN360-M* which uses our segmentation masks and SPIn-NeRF's inpainting technique. While SPN360-M outperforms SPN-360, it still falls short of our InNeRF360's performance. Moreover, it is less effective than NeRFactor combined with $\mathcal{L}_{\text{geom}}$, see results in Tab.1 of the main paper. To show the reliability of our numerical results, we provide the requested evaluation for *Ideal*, i.e., fitting a NeRF on ground truth scenes without the object(s) to be inpainted, which serves as an upper bound for the best inpainting results with the same inputs and NeRF architecture.

We capture ground truth datasets with the objects removed from the scene, with which we evaluate LPIPS [44] and Frechet Inception Distance (FID) [11] metric as metric. As indicated in Tab. 1 and Tab. 3, InNeRF360 outperforms the other two methods in terms of the similarity between the activations of the inpainted region and the ground truth.

Ablation on mask dilation. In Fig. 11, we demonstrate the importance of mask dilatation. We conducted experiments using different numbers of pixels on the contour for dilation, specifically with the set $\{11, 21, 41, 51, 101\}$, and determined that 51 pixels yield the best inpainting performance. Allowing for more contextual information in the 2D image inpainter encourages the inclusion of more background details, thereby enhancing consistency across different views. However, due to the limited 3D understanding of 2D image inpainters, further refinement of the initially inpainted scene is necessary.

Methods	Starbucks		Glass Cat		Multiple	
	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓
Ideal	0.4016	130.79	0.3928	129.95	0.3829	124.82
ObjNeRF-M	0.6967	275.92	0.7048	288.91	0.6743	260.48
SPN360-M	0.6037	198.64	0.6542	253.74	0.6073	192.45
InNeRF360	**0.4523**	**153.46**	**0.4158**	**142.60**	**0.4264**	**147.49**

Table 3. Quantitative results of requested baselines Ideal, ObjectNeRF-M, SPN-360-M and our InNeRF360 on rw datasets.

Methods	LPIPS ↓	FID ↓
SPIn-NeRF	0.4971	149.41
Ours	**0.4764**	**129.54**

Table 4. Quantitative comparison with SPIn-NeRF on the quality of synthesized inpainting regions, averaged over the front-facing datasets that we evaluate.

Front-facing scenes. InNeRF360 works for front-facing scenes with only text instructions, and without hand-drawn masks as SPIn-NeRF. In Fig. 12, our method synthesizes inpainting content that is more perceptually consistent with the surroundings for the staircase without introducing artifacts, while SPIn-NeRF leaves the partial shadow of the box. On the bench scene, our method also gives a more perceptually robust synthesized texture to the fence. Fig. 13 show additional qualitative results of InNeRF360 on frontal datasets. Our method does not introduce visual artifacts to the inpainted regions, and we encourage the reader to inspect our supplementary video for better visualization.

For quantitative analysis, we evaluate our method on the SPIn-NeRF frontal datasets by comparing the synthesized contents in the bounding box region with the provided ground truth images, following the setup of SPIn-NeRF. As displayed in, Tab. 4, InNeRF360 synthesizes content that is closer to the ground truth data.

B. Dataset Details

We conducted experiments on ten 360-degree scenes from various datasets: *Bear*, *Vasedeck*, *Garden*, *Room*, *Bulldozer* and *Floating Tree*, as well as four captured scenes, Cup, Starbucks, Glass Cat and 4-Objects specif-

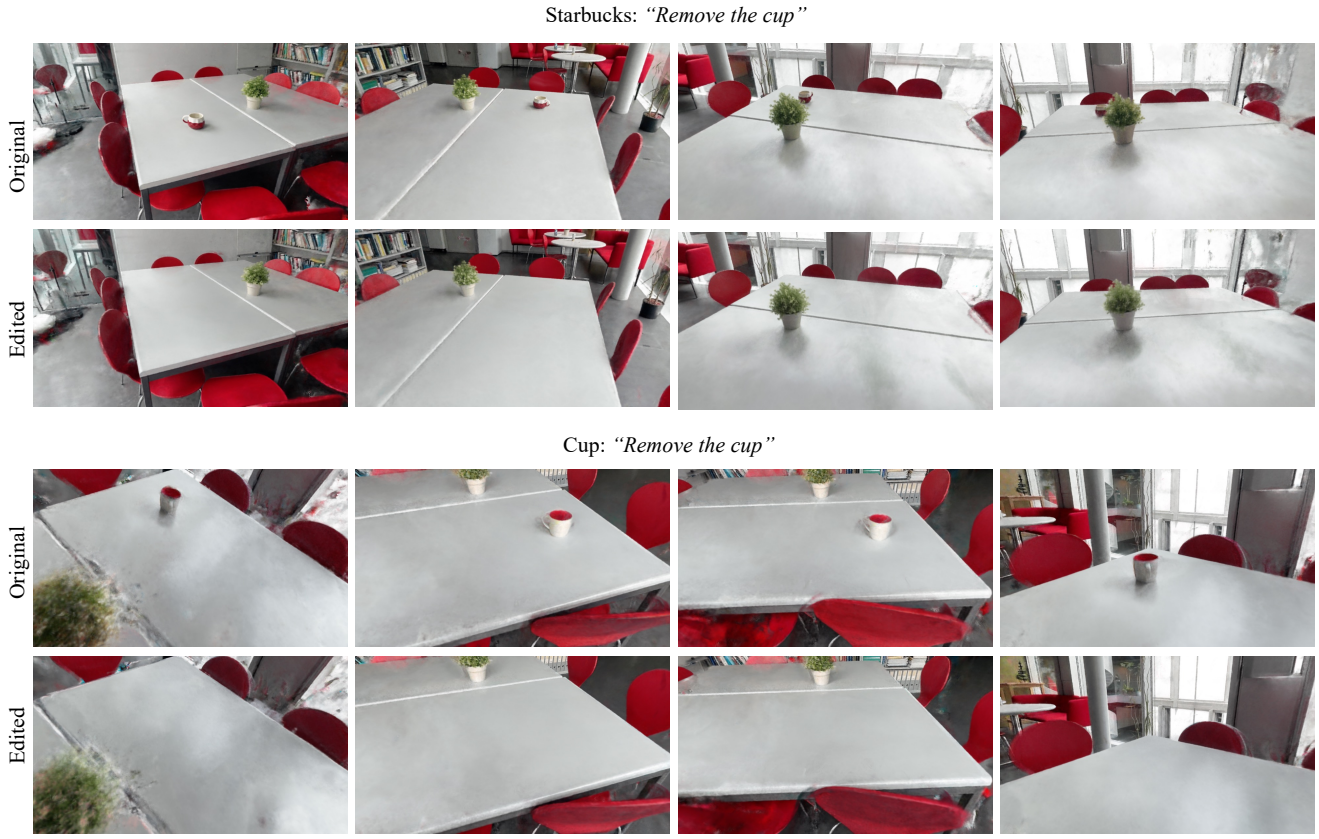


Figure 10. **Quantitative results on our captured datasets.** InNeRF360 generates consistent inpainting region on the occluded sections between the plant and the cup.

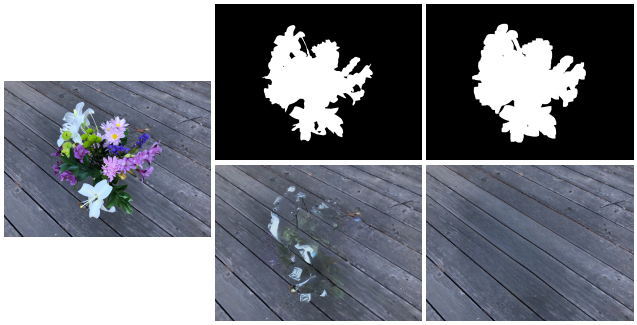


Figure 11. Contexualized segmentation. (a) Original image; (b) top: segmentation masks *without* dilation; bottom: resulting inpainted image; (c) top: segmentation masks *with* dilation; bottom: resulting inpainted image. Dilating the mask provides contextual pixel information and improves inpainting quality.

ically for quantitative evaluation purposes. We select 360° scenes containing different challenging aspects for the segmentation and inpainting tasks. *Bear* contains a large section to remove and complex background texture; *Vasedeck* contains a transparent object to select; *Room* contains multiple objects in different places in the scene; *Bulldozer* contains multiple instances of occlusion between objects to remove and other objects in the scene; *Floating Tree* contains object that does not locate on a flat surface.

Our scenes are captured using a smartphone, and the camera poses are extracted using PolyCam [32]. It's important to note that the estimated camera poses are object-centric but may contain noise, which can result in blurriness outside of the object region. Consequently, for our quantitative evaluation, we specifically focus on assessing the inpainting performance within the bounding box regions of the objects. The number of images included in each scene can be found in Tab. 5. The camera poses are sampled in the open area above and around the object(s) to be inpainted.

C. User Studies on View Consistency

We evaluate the user studies as shown in Tab. 2 with 49 users. For each scene, we present each participant with two 95-frame video clips: one rendered from our inpainted NeRF scene and the other from per-frame inpainted renderings of the original NeRF scene. Participants are asked to indicate which video appears more visually consistent and perceptually plausible. We calculate the percentage pref-

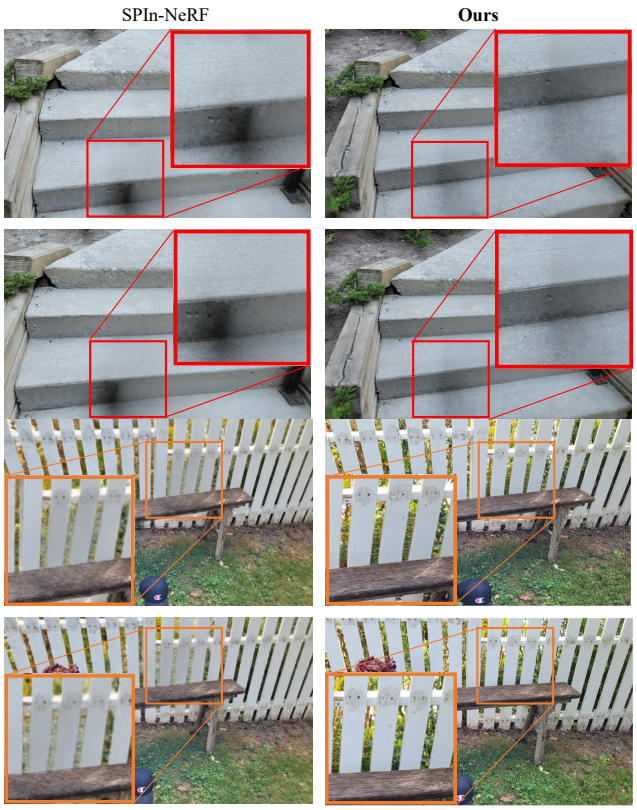


Figure 12. Qualitative comparison with SPIn-NeRF on front-facing datasets. InNeRF360 generates 3D-consistent inpaintings that contain fewer visual artifacts and are more aligned with surrounding regions.

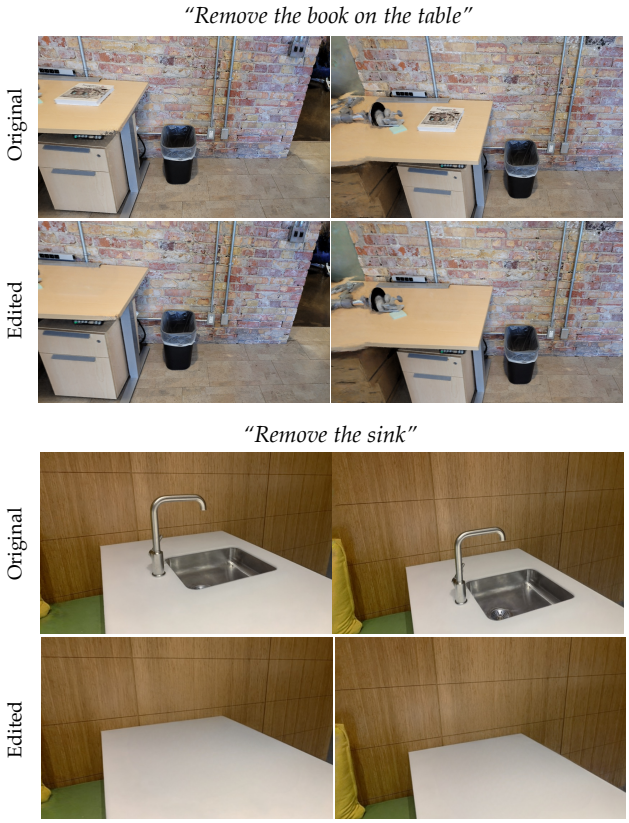


Figure 13. Qualitative results of InNeRF360 on frontal dataset scenes. More results can be found in our supplementary video.

360°	Size	Captured	Size	Frontal	Sizes
Vasedeck	116	Cup	117	[26]-(10)	60
Garden	185	Starbucks	199	Book	60
Room	311	4-Objects	206	Sink	60
Bulldozer	359	Glass Cat	136	Stairs	60
Bear	96			[43]-(001)	260
Floating Tree	96				

Table 5. Number of images in each dataset.

erence for each option by dividing the number of votes by the total number of participants. Fig. 14 provides a set of selected examples that we provide to the users.

This experiment aims to demonstrate that InNeRF360 offers **superior view-consistency** across frames compared to per-frame inpaintings. InNeRF360 achieves such performance due to our geometric and appearance refinement to the initialized NeRF from 2D inpainting. Our approach includes a geometric prior that works in 3D to remove density artifacts, and a masked LPIPS loss that ensures the inpainted region blends perceptually consistently with surrounding areas during training. Additionally, a trained NeRF scene inherently maintains 3D consistency. In contrast, per-frame editing relies solely on the local information of a single 2D viewpoint, lacking 3D consistency across different views. This can result in visual artifacts, which InNeRF360 effectively addresses and resolves.

D. Editing Accuracy

With a simple modification of our method by replacing the image inpainter with a mask-conditioned image editor [8], our method can produce view-consistent editing on specific objects instructed by text, as shown in Fig. 15. In this example, we do not use our $\mathcal{L}_{\text{geom}}$. Our baseline Instruct-NeRF2NeRF (In2n) [9] is also capable of stylizing NeRF scenes, but it cannot pinpoint a particular object for either removal or editing. In2n relies on Instruct-Pix2Pix [3] and operates solely in latent space for image editing. It applies stylization to a large undesired area of the NeRF scene, as illustrated in Fig. 9. In comparison, the modified version of InNeRF360 works with object-level modification, and delivers accurate editing results that accurately address the requested object. Specifically, we utilize our 3D consistent segmentation module to output masks for the dataset images. Then our method can be connected with

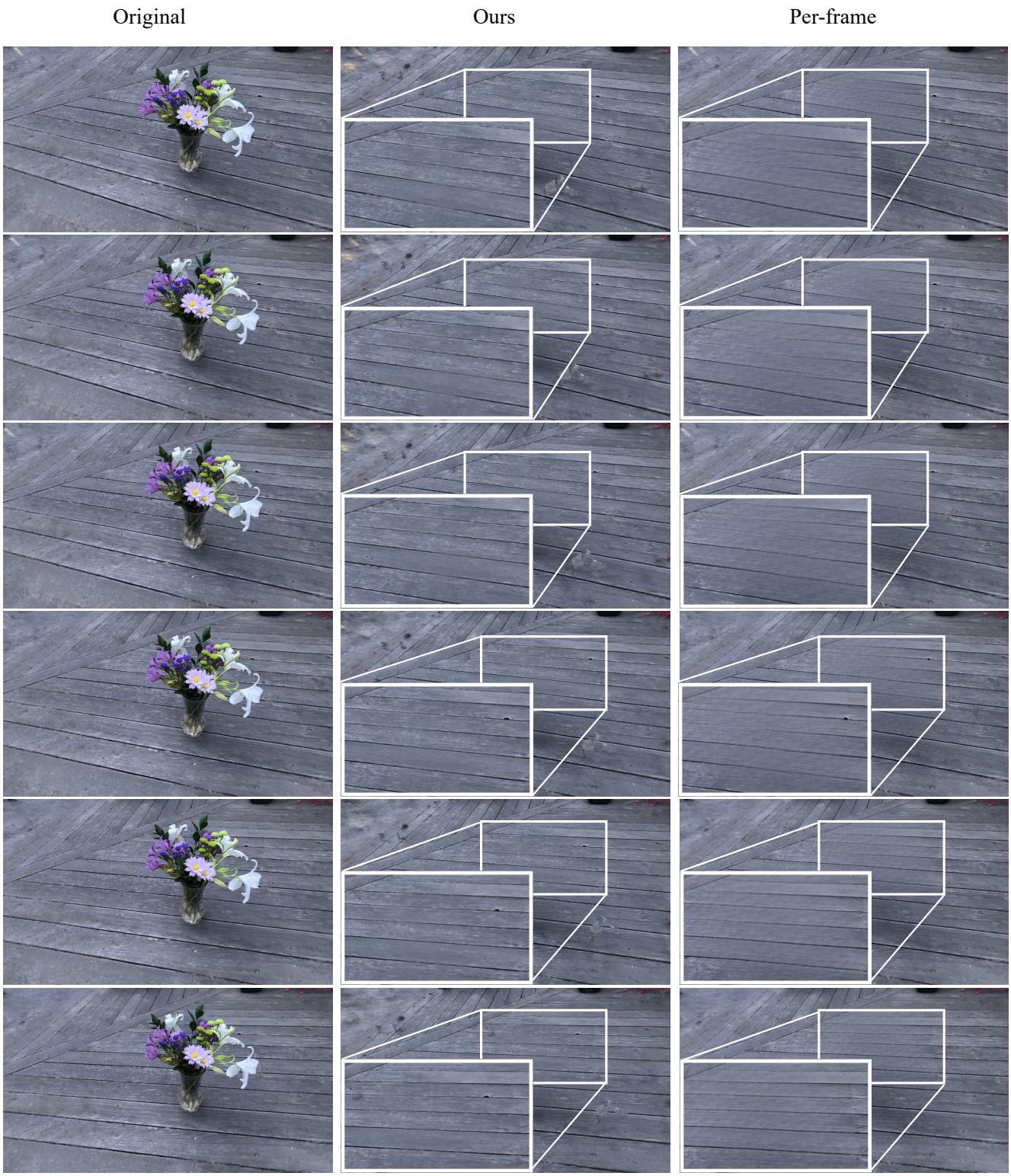


Figure 14. Qualitative comparison on view consistency across different camera poses between InNeRF360 and per-frame inpainting. The zoomed-in region in each frame shows the inpainted quality of our method and per-frame inpainting. Our method contains higher consistency across frames, while the per-frame inpainting tends to be inconsistent and blurry in the inpainted region.

Figure 15. Comparison on In2n with a modified version of InNeRF360 on object-level stylization.

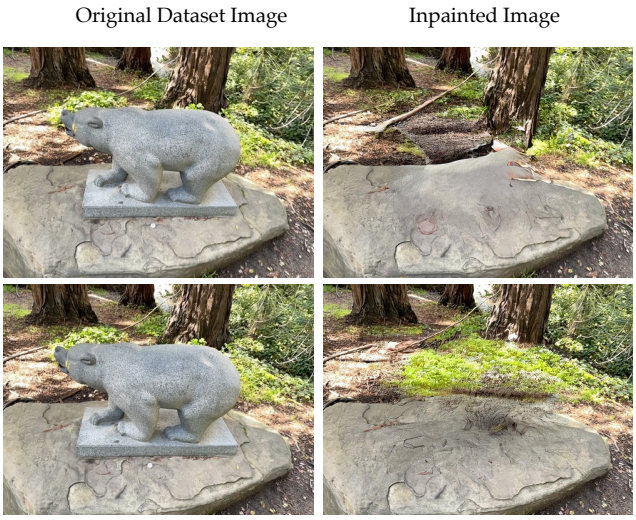


Figure 16. Examples of varying inpainted regions on the bear dataset images. We use the 2D image inpainter [36]. We can see the inconsistent inpaintings for two dataset images with similar camera origins, which can lead to concentrated artifacts near the inpainted region if trained into NeRF directly.

a mask-conditioned image editing method for object-level editing. Here we use [8], a zero-shot image editing method given a text-to-image denoising diffusion model.

Note that, however, the focus of InNeRF360 is to **remove** text-instructed object from the 360° NeRF scene. We provide such examples only to demonstrate the easily achievable extension to object-level editable NeRF with our proposed segmentation method, by making use of powerful 2D image processing tools to address challenging 3D problems.

E. 2D Inpainting Examples

In the scene of *Bear*, the object of interest takes up a large section of the image, and the background around the bear has non-uniform and highly varying patterns. In these cases, diffusion models tend to generate significant variations in pixel content for the inpainted region across different viewing perspectives, as indicated in Ln 385-386 of the main paper on examples of noisy 2D inpainted images. An example illustrating this limitation can be observed in Fig.16 (b), where the inpainted regions exhibit significant discrepancies between two adjacent viewpoints as shown in (a). There are also some results on per-frame inconsistency in our supplementary video.

F. Failure Cases

In *Vasedeck*: While the stain on the table is visually plausible and view-consistent, it is in fact the shadow of the object that was removed. Such shadow has soft edges and can easily be mistaken as part of the floor texture, and thus is overlooked by the image inpainter. Similarly in the staircase scene in frontal datasets, our method cannot completely remove soft shadow, although we produce more plausible results than the baseline method.

Figure 17. Additional segmentation results on *Bear*, *Garden* and *Vasedeck*.

G. Design Justification on Geometry Guidance

In InNeRF360, we opt not to use the inpainted depth images as inpainting priors on geometry like our prior works [25, 26, 43]. Inpainting on 2D depth maps creates inconsistency in geometry supervision between different views, similar to inpainting on RGB images in 2D. Instead, we utilize a trained 3D diffusion model as priors to super-

vise the removal of floaters, and therefore operate directly in 3D space to avoid inconsistency in geometry supervision.

On the other hand, we observe that inpainting depth maps enforce the accumulated floaters in the inpainted region to be "scattered" onto surrounding background environments, causing a blurry background similar to training with pixel-wise L1 in the inpainted region. As shown in Fig. 6, the background water pipe has been made very blurry in the output of SPN-360 which inpaints on both depth maps and RGB images. By not modifying 2D depth maps but operating on 3D space to remove density artifacts directly, we propose a method that is more effective than scattering such artifacts around into unconcentrated regions. The artifacts in the latter scenario are harder to resolve.

H. Implementation Details.

H.1 Training geometric priors

During the training on shapenet, we use voxelized cubes of $m^3 = 32 \times 32 \times 32$. We clamp the density values in voxel grids into $[0, 1]$. During inference time, the geometric prior performs one forward pass without backpropagating through the diffusion model. We set the threshold for determining whether a voxel is empty to be $\rho = 0.01$, and for floater detection and removal, we set $w = 0.02$. Looking at Eq. (6), we experiment with the value of w, which is a hyperparameter for the amount of density to be increased in the occupied voxels, and realize that it trades off with inpainting quality. Specifically, as we decrease w, more densities are guided by the diffusion priors to be removed, and thus occasionally we observe see-through surfaces in the trained scene. As we increase w, floaters artifact may not be completely removed from the scene. The learning rate t is chosen between 10 and 50. We empirically find out that for objects that take up small regions in the scene such as *Bulldozer*, the sampled cube size should be upper bounded around 5% of the scene. A more automatic way of determining the sampled cube size using the prompt and the Shapenet dataset will be interesting to explore in future work.

H.2 Depth warping refinement

For each training view, we select 8 other views with 20 points per view for depth-warping, for a balanced choice over efficiency on the iteration of dataset images and refined segmentation quality.

H.3 Training Inpainted NeRF

For the 2D image inpainter, we adapt the open-sourced code from the Latent Diffusion Model [36]. We use the 'Nerfacto' model from NeRFStudio [39] as our underlying backbone and adapt the implementation of diffusion priors training from Nerfbusters [42]. $\lambda_{\text{geom}} = 0.1$ and $\lambda_{\text{in}} = 0.1$. During training, we train for 1500 iterations without $\mathcal{L}_{\text{geom}}$ for initialization, and another 2500 iterations sampling 30 cubes per iteration.

H.4 Implementation details on SPN-360

We proposed a baseline SPN-360 that is stronger than directly adapting SPIn-NeRF [26] on 360° data, as its released implementation does not support 360-degree NeRF. The implementation of SPN-360 is based on NeRFacto. We obtain masks for SPN-360 through initialization from Dino and Semantic NeRF refinement, and inpaint with LaMa [38] on both RGB and depth map. Its segmentation masks and inpainting results are both generated by the method used in SPIn-NeRF so that we can compare them with our In-NeRF360 results

I. Additional results on segmentation.

Fig. 17 shows additional segmentation results on selected datasets with our segmentation module.

J. Acknowledgements

The authors thank Michele Vidulis and Desmond (Zhenyuan) Liu for their time spent proofreading and kind suggestions during the paper writing. The authors also thank the generous and insightful comments from all the reviewers, without whom this work will not be able to come to its current shape.