SPIn-NeRF: Multiview Segmentation and Perceptual Inpainting with Neural Radiance Fields

Ashkan Mirzaei[1,2] Tristan Aumentado-Armstrong[1,2,4] Konstantinos G. Derpanis[1,3,4]
Jonathan Kelly[2] Marcus A. Brubaker[1,3,4] Igor Gilitschenski[2] Alex Levinshtein[1]
[1]Samsung AI Centre Toronto [2]University of Toronto [3]York University [4]Vector Institute for AI
{a.mirzaei,tristan.a}@partner.samsung.com, {jkelly,gilitschenski}@cs.toronto.edu
{kosta,mab}@eecs.yorku.ca, alex.lev@samsung.com

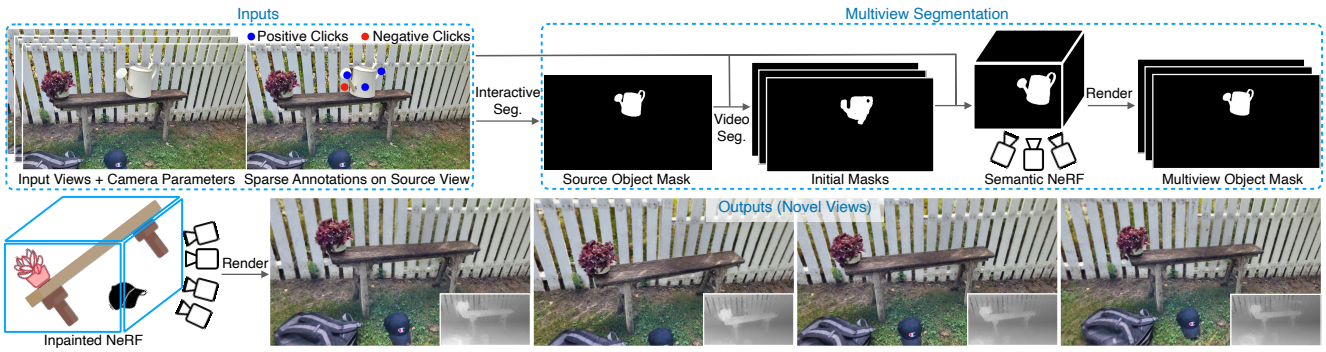

Figure 1. An example of the inputs and outputs of our 3D inpainting framework. In addition to the images captured from the scene and their corresponding camera parameters, users are tasked with providing a few points in a single image to indicate which object they wish to remove from the scene (upper-left inset). These sparse annotations are then automatically transferred to all other views, and utilized for multiview mask construction (upper-right inset). The resulting 3D-consistent mask is used in a perceptual optimization problem that results in 3D scene inpainting (lower row), with rendered depth from the optimized NeRF shown for each image as an inset.

Abstract

Neural Radiance Fields (NeRFs) have emerged as a popular approach for novel view synthesis. While NeRFs are quickly being adapted for a wider set of applications, intuitively editing NeRF scenes is still an open challenge. One important editing task is the removal of unwanted objects from a 3D scene, such that the replaced region is visually plausible and consistent with its context. We refer to this task as 3D inpainting*. In 3D, solutions must be both consistent across multiple views and geometrically valid. In this paper, we propose a novel 3D inpainting method that addresses these challenges. Given a small set of posed images and sparse annotations in a single input image, our framework first rapidly obtains a 3D segmentation mask for a target object. Using the mask, a perceptual optimization-based approach is then introduced that leverages learned 2D image inpainters, distilling their information into 3D space, while ensuring view consistency. We also address the lack of a diverse benchmark for evaluating 3D scene inpainting methods by introducing a dataset comprised of challenging real-world scenes. In particular, our dataset contains views of the same scene with and without a target object, enabling more principled benchmarking of the 3D inpainting task. We first demonstrate the superiority of our approach on multiview segmentation, comparing to NeRF-based methods and 2D segmentation approaches. We then evaluate on the task of 3D inpainting, establishing state-of-the-art performance against other NeRF manipulation algorithms, as well as a strong 2D image inpainter baseline. We encourage the readers to visit our project website.*

1. Introduction

Neural rendering methods, especially Neural Radiance Fields (NeRFs) [35], have recently emerged as a new modality for representing and reconstructing scenes [50], achieving impressive results for novel view synthesis. Substantial research effort continues to focus on formulating more efficient NeRFs (e.g., [6, 20, 43]), to make NeRFs

more accessible in use-cases with more limited computational resources. As NeRFs become more widely accessible, the need for editing and manipulating the scenes represented by NeRFs will continue to grow. One notable editing application is removing objects and inpainting the 3D scene, analogous to the well-studied 2D image inpainting task [23]. However, several obstacles impede progress on this task, not only for the 3D inpainting process itself, but also in obtaining the input segmentation masks. First, NeRF scenes are implicitly encoded within the neural mapping weights, resulting in an entangled and uninterpretable representation that is non-trivial to manipulate (compared to, say, the explicit discretized form of 2D image arrays or meshes in 3D). Moreover, any attempt to inpaint a 3D scene must not only generate a perceptually realistic appearance in a single given view, but also preserve fundamental 3D properties, such as appearance consistency across views and geometric plausibility. Finally, to obtain masks for the target object, it is more intuitive for most end users to interact with 2D images, rather than 3D interfaces; however, requiring annotations of multiple images (and maintaining view-consistent segments) is burdensome to users. An appealing alternative is to expect only a minimal set of annotations for a single view. This motivates a method capable of obtaining a view-consistent 3D segmentation mask of the object (for use in inpainting) from single-view sparse annotations.

In this paper, we address these challenges with an integrated method that takes in multiview images of a scene, extracts a 3D mask with minimal user input, and fits a NeRF to the masked images, such that the target object is replaced with plausible 3D appearance and geometry. Existing interactive 2D segmentation methods do not consider the 3D aspects of the problem (e.g., [42]), while current NeRF-based approaches are unable to use sparse annotations [75] to perform well, or do not attain sufficient accuracy [44]. Similarly, while some current NeRF manipulation algorithms allow object removal, they do not attempt to provide perceptually realistic inpaintings of newly unveiled parts of space (e.g., [63]). To our knowledge, this is the first approach that handles both interactive multiview segmentation and full 3D inpainting in a single framework.

Our technique leverages off-the-shelf, 2D-based, 3D-unaware models for segmentation and inpainting, and transfers their outputs to 3D space in a view-consistent manner. Building on the (2D) interactive segmentation [8, 15, 33] literature, our framework starts from a small number of user-defined image points on a target object (and a few negative samples outside it). From these, our segmentation algorithm initializes masks with a video-based model [4], and lifts them into a coherent 3D segmentation via fitting a semantic NeRF [36, 75, 76]. Then, after applying a pretrained 2D inpainter [48] to the multiview image set, a customized NeRF fitting process is used to reconstruct the 3D inpainted scene, utilizing perceptual losses [71] to account for inconsistencies in the 2D inpainted images, as well as inpainted depth images to regularize the geometry of the masked region. Overall, we provide a complete method, from object selection to novel view synthesis of the inpainted scenes, in a unified framework with minimal burden on the user, illustrated in Figure 1.

We demonstrate the effectiveness of our approach through extensive qualitative and quantitative evaluations. In addition, we address the lack of a benchmark for comparing scene inpainting methods, and introduce a new dataset where the "ground-truth inpaintings" (i.e., real images of the scene without the object) are available as well.

In summary, our contributions are as follows: (i) a complete process for 3D scene manipulation, starting from object selection with minimal user interaction and ending with a 3D inpainted scene in the form of a NeRF; (ii) to perform such selection, an extension of 2D segmentation models to the multiview case, capable of recovering 3D-consistent masks from sparse annotations; (iii) to ensure such outputs are view-consistent and perceptually plausible, a novel optimization-based formulation of 3D inpainting in NeRFs, which leverages 2D inpainters; and (iv) a new dataset for 3D object removal evaluation that includes corresponding object-free ground-truth. Our code and dataset will be made publicly available upon acceptance.

2. Related Work

Image Inpainting. Inpainting in 2D has received significant research attention [23]. While early techniques relied on patches [9, 16], recent neural methods optimize both perceptual realism and reconstruction (e.g., [22, 27, 48]). Various lines of research continue to be explored for improving visual fidelity, including adversarial training (e.g., [40, 72]), architectural advances (e.g., [27, 30, 61, 62]), pluralistic outputs (e.g., [72, 74]), multiscale processing (e.g., [21, 58, 68]), and perceptual metrics (e.g., [22, 48, 71]). Our work leverages LaMa [48], which applies frequency-domain transforms [7] inspired by transformers [11, 12]. Yet, none of these approaches lift the problem into 3D; thus, inpainting multiple captures of a scene in a consistent manner remains an underexplored task. While there are some existing 3D-aware image inpainting algorithms, they either only partially operate in 3D [64], rely on reference images [73], or consider more limited scenarios [24]. In contrast, our method operates directly in the 3D domain, via the multiview-based NeRF model.

NeRF Manipulation. Representing scenes via volumetric rendering has recently become an important research direction [50]. Based on differentiable volume rendering [18, 52] and positional encoding [13, 49, 53], NeRFs [35] have demonstrated remarkable performance in novel-view synthesis. Recent works have studied potential improve-

ments in NeRF's training or rendering speed [5, 6, 17, 38, 47, 66], reconstruction quality [1, 2, 10, 29], and data requirements [28, 56, 59, 67]. However, manipulating NeRF scenes remains a challenge. Clip-NeRF [55], Object-NeRF [63], LaTeRF [36], and others [25, 32, 69] introduce approaches to alter and complete objects represented by NeRFs; however, their performance is limited to simple objects, rather than scenes with significant clutter and texture, or they focus on tasks other than general inpainting (e.g., recoloring or deforming). Most closely related to our method is NeRF-In [31], a concurrent unpublished work, which inpaints NeRF scenes with geometry and radiance priors from 2D image inpainters, but does not address inconsistency; instead, its use of a simple pixelwise loss relegates it to simply reducing the number of views used for fitting, which reduces final view synthesis quality. In contrast, our approach is able to inpaint NeRF representations of challenging real-world scenes by incorporating 2D information in a view-consistent manner, via a *perceptual* [71] training regime. This avoids over-constraints on the inpainting, which would normally lead to blurriness.

3. Background: Neural Radiance Fields

NeRFs [35] encode a 3D scene as a function, $f : (x, d) \rightarrow (c, \sigma)$, that maps a 3D coordinate, x, and a view direction, d, to a color, c, and density, σ. The function f can be modelled in various ways (e.g., [35, 47]). For a ray, r, the expected color is estimated by volumetric rendering via quadrature; the ray is divided into N sections between t_n and t_f (the near and far bounds), with t_i sampled from the i-th section, to render the estimated color, $\widehat{C}(r)$:

$$\widehat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))c_i, \tag{1}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$ is the transmittance, $\delta_i = t_{i+1} - t_i$ is the distance between two adjacent points, and c_i and σ_i are the color and density at t_i, respectively. For the rays passing through pixels of the training views, the ground-truth color $C_{\text{GT}}(r)$ is available, and the model is optimized using the reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \sum_{r \in \mathcal{R}} \|\widehat{C}(r) - C_{\text{GT}}(r)\|^2, \tag{2}$$

where $\mathcal{R}$ is a ray batch sampled from the training views.

4. Method

Given a set of RGB images, $\mathcal{I} = \{I_i\}_{i=1}^n$, with corresponding 3D poses, $\mathcal{G} = \{G_i\}_{i=1}^n$, and camera intrinsic matrix, K, our model expects one additional "source" view with sparse user annotations (i.e., a few points identifying the unwanted object). From these inputs, we produce a NeRF model of the scene, capable of synthesizing an *inpainted* image from any novel view. We begin by obtaining an initial 3D mask from the single-view annotated source (§ 4.1.1), followed by fitting a semantic NeRF, to improve the consistency and quality of the mask (§ 4.1.2). Finally, in § 4.2 we describe our view-consistent inpainting method, which takes the views and recovered masks as inputs. Our approach leverages the outputs of 2D inpainters [48] as appearance and geometry priors to supervise the fitting of a new NeRF. Figure 1 illustrates our entire approach, including the inputs and outputs. Additional details are in our supplementary material.

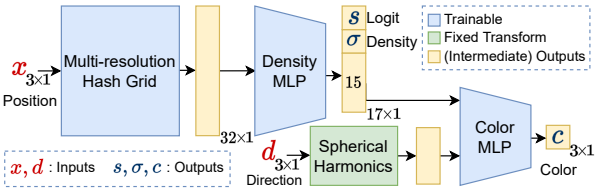

Figure 2. Overview of our multiview segmentation architecture. As input, this network takes in a 3D coordinate, x, and a view direction, d, and returns view-independent density, $\sigma(x)$, objectness logit, $s(x)$, and view-dependent color, $c(x, d)$.

4.1. Multiview Segmentation

4.1.1 Mask Initialization

We first describe how we initialize a rough 3D mask from single-view annotations. Denote the annotated source view as I_1. The sparse information about the object and the source view are given to an interactive segmentation model [15] to estimate the initial source object mask, $\widehat{M}_1$. The training views are then treated as a video sequence and, along with $\widehat{M}_1$, given to a video instance segmentation model V [4, 57], to compute $V(\{I_i\}_{i=1}^n, \widehat{M}_1) = \{\widehat{M}_i\}_{i=1}^n$, where $\widehat{M}_i$ is the initial guess for the object mask for I_i. The initial masks, $\{\widehat{M}_i\}_{i=1}^n$, are typically inaccurate around the boundaries, since the training views are not actually adjacent video frames, and video segmentation models are usually 3D-unaware. Hence, we use a semantic NeRF model [36, 75, 76] to resolve the inconsistencies and improve the masks (§ 4.1.2), thus obtaining the masks for each input view, $\{M_i\}_{i=1}^n$, to use for inpainting (§ 4.2).

4.1.2 NeRF-based Segmentation

Our multiview segmentation module takes the input RGB images, $\{I_i\}_{i=1}^n$, the corresponding camera intrinsic and extrinsic parameters, and the initial masks, $\{\widehat{M}_i\}_{i=1}^n$, and trains a semantic NeRF [75]. Figure 2 depicts the network used in the semantic NeRF; for a point, x, and a view direction, d, in addition to a density, $\sigma(x)$, and color, $c(x, d)$, it

returns a pre-sigmoid "objectness" logit, $s(x)$. The objectness probability is then acquired as $p(x) = \text{Sigmoid}\big(s(x)\big)$. We use Instant-NGP [3, 37, 38] as our NeRF architecture due to its fast convergence. The expected objectness logit, $\widehat{S}(r)$, associated with a ray, r, is obtained by rendering the logits of the points on r instead of their colors, with respect to the densities, in Eq. 1 [36]:

$$\widehat{S}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i)) s_i, \tag{3}$$

where for simplicity, $s(r(t_i))$ is denoted by s_i. The objectness probability of a ray, $\widehat{P}(r) = \text{Sigmoid}\big(\widehat{S}(r)\big)$, is then supervised using the classification loss:

$$\mathcal{L}_{\text{clf}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \text{BCE}\big(\mathbb{1}_{r \in \mathcal{R}_{\text{masked}}}, \widehat{P}(r)\big), \tag{4}$$

where $\mathbb{1}$ is the indicator function, BCE stands for the binary cross entropy loss, and $\mathcal{R}_{\text{masked}}$ is the set of rays passing through pixels that are masked in $\{\widehat{M}_i\}_{i=1}^n$. During the calculation of the classification loss, $\mathcal{L}_{\text{clf}}$, the weights of the colors in the rendering equation (Eq. 1) are detached to limit the supervised updates to the logits; this prevents changes to the existing geometry, due to gradient updates altering the σ field. The geometry is supervised using a reconstruction loss, $\mathcal{L}_{\text{rec}}$, as in NeRF [35], via the given RGB images. The overall loss, used to supervise the NeRF-based multiview segmentation model, is then given by:

$$\mathcal{L}_{\text{mv}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{clf}} \mathcal{L}_{\text{clf}}, \tag{5}$$

where the classification weight, λ_{clf}, is a hyper-parameter. After optimization, 3D-consistent masks, $\{M_i\}_{i=1}^n$, are obtained by thresholding the objectness probabilities and masking the pixels with probabilities greater than 0.5. Finally, we use two stages for optimization to further improve the masks; after obtaining the initial 3D mask, the masks are rendered from the training views, and are used to supervise a secondary multiview segmentation model as initial guesses (instead of the video segmentation outputs).

4.2. Multiview Inpainting

Figure 3 shows an overview of our view-consistent inpainting method. As the paucity of data precludes directly training a 3D inpainter, our method leverages existing 2D inpainters to obtain depth and appearance priors, which then supervise the fitting of a NeRF to the completed scene. This inpainted NeRF is trained using the following loss:

$$\mathcal{L}_{\text{inp}} = \mathcal{L}'_{\text{rec}} + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}}, \tag{6}$$

where $\mathcal{L}'_{\text{rec}}$ is the reconstruction loss for the unmasked pixels, and $\mathcal{L}_{\text{LPIPS}}$ and $\mathcal{L}_{\text{depth}}$ define the perceptual and depth losses (detailed below), with hyper-parameter weights λ_{LPIPS} and λ_{depth}.

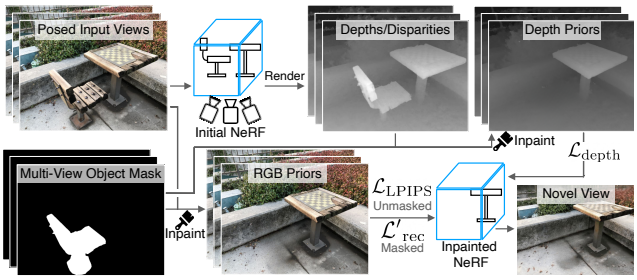

Figure 3. Overview of our inpainting method. Using posed input images and their corresponding masks (upper- and lower-left insets), we obtain (i) an initial NeRF *with* the target object present and (ii) the set of inpainted input RGB images with the target object removed (but with view inconsistencies). The initial NeRF (i) is used to compute depth, which we inpaint to obtain depth images as geometric priors (upper-right inset). The inpainted RGB images (ii), which act as appearance priors, are used with the depth priors, to fit a 3D consistent NeRF to the inpainted scene.

4.2.1 RGB Priors

Our proposed view-consistent inpainting approach uses RGB inputs, $\{I_i\}_{i=1}^n$, the camera intrinsic and extrinsic parameters, and corresponding object masks, $\{M_i\}_{i=1}^n$, to fit a NeRF to the scene without the undesired object. To begin with, each image and mask pair, (I_i, M_i), is given to an image inpainter, INP, to obtain the inpainted RGB images, $\{\widetilde{I}_i\}_{i=1}^n$, where $\widetilde{I}_i = \text{INP}(I_i, M_i)$ [48]. Since each view is inpainted independently, directly supervising a NeRF using the inpainted views leads to blurry results due to the 3D inconsistencies between each $\widetilde{I}_i$ (see Figure 7). In this paper, instead of using mean squared error (MSE) to optimize the masked area, we propose the use of a perceptual loss [71] to optimize the masked parts of the images, while still using MSE for the unmasked parts, where no inpainting is needed. The perceptual loss is calculated as follows:

$$\mathcal{L}_{\text{LPIPS}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \text{LPIPS}(\widehat{I}_i, \widetilde{I}_i), \tag{7}$$

where $\mathcal{B}$ is a batch of indices between 1 and n, LPIPS is the perceptual loss [71], and $\widehat{I}_i$ is the i-th view rendered using NeRF. Our model for multiview inpainting and segmentation uses the same architecture (shown in Figure 2), except for the additional logit output, s.

4.2.2 Depth Priors

Even with the perceptual loss, the discrepancies between the inpainted views can incorrectly guide the model towards converging to degenerate geometries (e.g., 'foggy' geometry may form near the cameras, to explain the disparate per-view information). Thus, we use inpainted depth maps as additional guidance for the NeRF model, and detach the

weights when calculating the perceptual loss and use the perceptual loss to only fit the colors of the scene. For this purpose, we use a NeRF optimized on images that include the unwanted object, and render the depth maps, $\{D_i\}_{i=1}^n$, corresponding to the training views. Depth maps are calculated by substituting the distance to the camera instead of the color of points into Eq. 1:

$$D(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))t_i. \tag{8}$$

The rendered depths are then given to an inpainter to obtain inpainted depth maps, $\{\widetilde{D}_i\}_{i=1}^n$, where $\widetilde{D}_i$ is obtained as $\widetilde{D}_i = \text{INP}(D_i, M_i)$. We found that using LaMa [48] for depth inpainting, as in the RGB case, gave sufficiently high-quality results. Note that this is all calculated as a preprocessing step, and with a NeRF optimized on the original scene. This NeRF can be the same model used for multiview segmentation. If using another source for obtaining masks, such as human annotated masks, a new NeRF is fitted to the scene. These depth maps are then used to supervise the inpainted NeRF's geometry, via the ℓ_2 distance of its rendered depths, $\widehat{D}_i$, to the inpainted depths, $\widetilde{D}_i$:

$$\mathcal{L}_{\text{depth}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \left| \widehat{D}(r) - \widetilde{D}(r) \right|^2, \tag{9}$$

where $\widehat{D}(r)$ and $\widetilde{D}(r)$ are the depth values for a ray, r.

4.2.3 Patch-based Optimization

Calculating the perceptual loss, Eq. 7, requires full input views to be rendered during the optimization. Since rendering each pixel necessitates multiple forward passes through the MLP, for high-resolution images, this is an expensive process, resulting in issues such as (i) the batch size, $|\mathcal{B}|$, has to be small to fit the rendered images and their corresponding computation graphs in memory, and (ii) slow optimization, even with batch sizes as small as $|\mathcal{B}| = 1$. A straightforward solution is to render a downsized image and compare it to the downsized version of the inpainted images; however, this leads to a loss of information if the downsizing factor is large. Following image-based works (e.g., SinGAN [46] and DPNN [14]), and 3D works (e.g., ARF [70]), we perform the computations on a patch-basis; instead of rendering complete views, we render batches of smaller patches, and compare them with their counterparts in the inpainted images based on the perceptual loss. Only patches inside the bounding box of the object mask are used. For fitting the unmasked areas, recall that $\mathcal{L}'_{\text{rec}}$ (Eq. 6) simply alters $\mathcal{L}_{\text{rec}}$ (Eq. 2) to sample rays only from unmasked pixels. By separating the perceptual and reconstruction losses, we prevent inconsistency within the mask, while avoiding unnecessary changes to the rest of the scene.

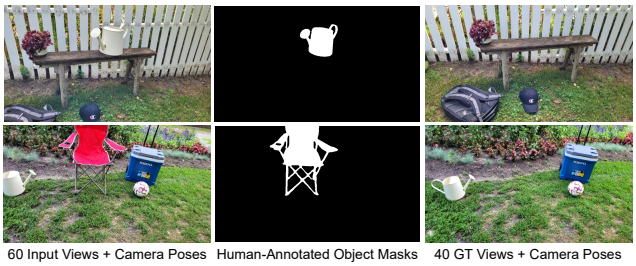

Figure 4. Sample scenes from our dataset. Columns: input view (left), corresponding target object mask (middle), and a ground-truth view without the target object, from a different camera pose (right). Rows: different scenes; see supplement for samples of all scenes in our dataset.

4.2.4 Mask Refinement

Here, we consider further leveraging the multiview data to guide the image inpainter. In particular, parts of the training images that are currently being generated by the 2D image inpainter might be visible in other views; in such cases, there is no need to hallucinate those details, since they can be retrieved from the other views. To prevent such unnecessary inpaintings, we propose a mask refinement approach: for each source image, depth, and mask tuple, (I_s, D_s, M_s), we substitute pixels in I_s and D_s that are visible from at least one other view, to shrink the source mask, M_s. After this refinement step, only parts of I_s and D_s that are occluded by the undesired object in *all* of the training views will remain masked. As a result, the image inpainter has to fill in a smaller area, resulting in improved inpaintings. Please see our supplementary material for details.

5. Experiments

Dataset. To evaluate multiview segmentation (MVSeg), we adopt real-world scenes from LLFF [34], NeRF-360 [35], NeRF-Supervision [65], and Shiny [60]. For multiview (MV) inpainting, in addition to providing qualitative results on scenes from IBRNet [56], we address the need for a standard benchmark including ground-truth captures of scenes without the unwanted object as test views, and introduce a dataset containing 10 real-world scenes with human annotated object masks. For each scene, we provide 60 training images with the object and 40 test images without the object (each image is 2268×4032 pixels in size). This dataset is further suitable for evaluating tasks such as real-time 3D inpainting, unsupervised 3D segmentation, and video inpainting. Figure 4 shows sample views from two scenes of our dataset.

Metrics. To evaluate our segmentation model, we use the accuracy of the predictions (pixel-wise) and the intersection over union (IoU) metric. For MV inpainting, we follow the image-to-image literature [48] and report the average

Table 1. Quantitative evaluation of our multiview segmentation model against the baselines for the task of transferring the source mask to other views.

Method	Acc.↑	IoU↑
Proj. + Grab Cut [45] (2D)	91.08	46.61
Proj. + EdgeFlow [15] (2D)	96.84	81.63
Semantic NeRF [75] (only source mask)	94.63	75.13
Proj. + EdgeFlow [15] + Semantic NeRF [75]	97.26	83.95
Feature Field Distillation [26]	97.37	83.07
Video Segmentation [4, 57]	98.43	88.34
Ours	98.85	90.96
Ours (two-stage)	**98.91**	**91.66**

learned perceptual image patch similarity (LPIPS) [71], and the average Fréchet inception distance (FID) [19] between the distribution of the ground-truth test views and model outputs. Since our focus is the inpainting, we only calculate the LPIPS and FID inside the bounding box of the object mask (using our MVSeg model, we can render the object mask from the test views that do not contain the object).

Multiview Segmentation Baselines. For MVSeg, one category of baselines we use for evaluations is projection-based approaches: the source mask is projected into the other views using the scene geometry from a NeRF. This gives us an incomplete mask in the other views. Then, a variety of interactive segmentation approaches are applied to the incomplete masks, propagating them to obtain complete object masks: *Proj. + Grab Cut [45]* and *Proj. + EdgeFLow [15]*. In addition, we consider *Proj. + EdgeFlow + Semantic NeRF*, where an additional Semantic NeRF [75] is fitted to make the outputs 3D consistent. Another baseline [26] is a representative of the concurrent works on distilling 2D pixel-level features to 3D scenes [51] and post-processing them for obtaining segmentation masks. As a baseline for video-segmentation [57], we compare to Dino [4] since it does not rely on temporally close neighboring frames.

Multiview Inpainting Baselines. Masked NeRF only uses the unmasked pixels to optimize a NeRF. Object NeRF [63] filters the unwanted points in 3D without explicitly inpainting the missing regions. No code is available for NeRF-In [31], so we use our own implementation of their model, with slight modifications (we use LaMa [48] as the inpainter). In addition, we compare our results to LaMa [48] as a representative of state-of-the-art 2D inpainters. To enable fair comparison between the NeRF-based 3D models (which, in addition to inpainting, have to synthesize novel views), we compare to LaMa by (i) fitting a NeRF on the views *with* the object[1], (ii) rendering the test views from the fitted NeRF, and (iii) passing these rendered images to LaMa. Finally, for reference and as an ideal "gold standard" 3D inpainting baseline, we fit a NeRF on the *ground-truth* test images, use the optimized NeRF to render the test-views, and then compare the rendered results to the ground-truth. We provide these results for completeness, as an upper-bound on the best possible results one can expect to obtain when using the same NeRF architecture.

[1]Since the test views that do not contain the object should not be available to the model during the inference.

Table 2. Quantitative evaluation of our inpainting method, using human-annotated object masks.

Method	LPIPS↓	FID↓
Ideal	0.4079	100.25
LaMa (2D) [48]	0.5369	174.61
Object NeRF [63]	0.6829	271.80
Masked NeRF [35]	0.6030	294.69
NeRF-In [31]	0.5699	238.33
NeRF-In [31] (Single)	0.4884	183.23
Ours (no geo. inpainting)	0.4952	200.34
Ours	0.4654	156.64
Ours (Refined RGB/Depth)	0.4664	163.79
Ours (Refined RGB)	**0.4529**	**147.31**

5.1. Results

Multiview Segmentation. We first evaluate our MVSeg model without any inpainting. In this experiment, we assume that the sparse image points are already given to an off-the-shelf interactive segmentation model, and that the source mask is available. Thus, the task is to transfer the source mask to other views. Table 1 shows that our model outperforms all of the 2D (3D-inconsistent) and 3D-consistent baselines. In addition, our two-stage optimization helps to further improve the obtained masks.

Qualitatively, Figure 5 compares the results of our segmentation model with the outputs of Neural Volumetric Object Selection (NVOS) [44] and video segmentation [4, 57]. Compared to the coarse edges of the 3D-unaware video segmentation model, our model reduces noise and improves consistency across views. Although NVOS uses scribbles, as opposed to the sparse points used in our model, our MVSeg visually outperforms NVOS. Since the NVOS code base is not available, we reproduce the published qualitative results for NVOS [44]. For additional examples, please see our supplementary material.

Multiview Inpainting. Our dataset is used for quantitative evaluation of our proposed inpainting method against the baselines. The object masks are slightly dilated (for five iterations with a 5×5 kernel) to reduce the effects of the shadow of the target objects in the inpainted scene and to make sure that the mask covers all of the object. Table 2 shows the comparison of our MV inpainting method with the baselines, assuming that the object masks from all of the views are given. The "ideal" row is not a baseline, but rather a NeRF fitted to the ground-truth test views (views

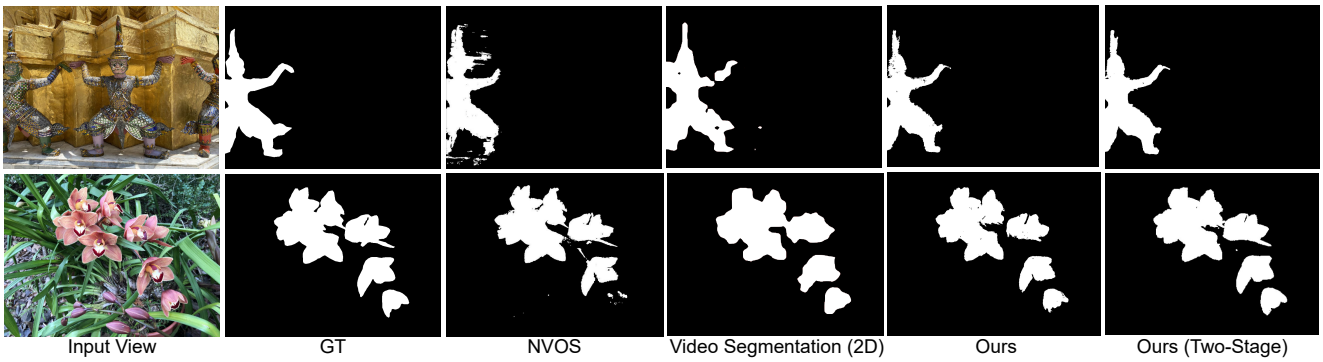

Figure 5. A qualitative comparison of our multiview segmentation model against Neural Volumetric Object Selection (NVOS) [44] (reproduced from original paper), video segmentation [4], and the human-annotated masks (GT). Our two-stage approach refers to running our multiview segmentation model twice, with the output of the first run as the initialization of the second (see § 4.1.2). Our method is less noisy than NVOS, which also misses some pieces of the target (e.g., the lowest flower in the bottom row), but captures more details than the video segmentation alone (e.g., the blurred edges of the flowers). Our two-stage approach helps fill in some missing aspects of the one-stage output. Please see our supplement for additional qualitative examples.

Table 3. Quantitative evaluation of our inpainting method, using the outputs of our multiview segmentation model.

Method	**LPIPS**↓	**FID**↓
LaMa (2D) [48]	0.5439	169.92
Object NeRF [63]	0.6679	286.55
Masked NeRF [35]	0.6340	332.70
NeRF-In [31]	0.5858	240.27
NeRF-In [31] (Single)	0.5054	176.27
Ours	**0.4662**	**140.56**

of the scene without the object). While this is only one instantiation of the many possible constructions of a plausible inpainted scene, it provides a convenient measure of the best performance one might reasonably expect in this scenario. Overall, our method significantly outperforms the alternative 2D and 3D inpainting approaches. Although our model uses a 2D image inpainter's [48] outputs to obtain a view-consistent inpainted NeRF, it is able to use this ensemble of MV information, in addition to the priors encoded by the perceptual loss function, to outperform the 2D inpainter. Table 2 further shows that the model without any geometry guidance has difficulty finding the correct geometry of the inpainted scene from inconsistent views.

We display qualitative results of our MV inpainting method in Figure 6, showing that it can reconstruct a view-consistent scene with detailed textures, including coherent view-dependent radiance for both shiny and non-shiny surfaces. In addition, in Figure 7, we provide a visual comparison to the unpublished concurrent work NeRF-In [31], which has the second-lowest error. We observe that a NeRF-In model fitted to all of the inpainted views results in blurry outputs. Alternatively, using a single inpainted view for supervising the masked region leads to artifacts in further views, due to the lack of supervision for the view-dependent radiance, as well as the poor extrapolation capabilities of the network. In contrast, our perceptual approach relaxes the exact reconstruction constraints in the masked region, thus preventing blurriness despite using all of the images, while avoiding artifacts caused by single-view supervision.

Table 2 shows that refinement provides a small but significant boost in inpainting quality, due to smaller masks requiring less hallucination from the inpainter. Yet, empirically, refinement only subtly trims the masks (reducing mean masked area by 4.74% on our dataset), as the camera has limited movement during data collection, to ensure similarity between the training and testing views. Further, due to noisy NeRF geometries projecting incorrect values, refining *depths* lowers performance, whereas refining *colours alone* achieves our best results.

So far, our experiments examine the performance of our MVSeg and MV inpainting independently; however, one can combine them to remove objects from NeRF scenes with minimal user interaction. Table 3 shows that using the output masks of our MVSeg model, instead of using the human-annotated object masks, results in a subtle decrease in the inpainting quality. However, our model with MVSegs still outperforms other methods, even when they are fitted on *human*-annotated segmentations.

5.2. Variations and Ablation Studies

Number of Input Views. Limiting the number of input views is a standard approach employed in the literature to modulate the reconstruction quality of NeRFs [54]. Table 4 shows that the performance of our inpainter degrades with fewer inputs. Thus, we argue that as better-quality NeRFs are introduced, our approach, which is agnostic to the underlying NeRF model, can readily benefit.

Figure 6. Visualizations of our view-consistent inpainting results. Upper rows per inset show NeRF renderings of the original scene from novel views, with the first image also displaying the associated mask. Lower rows show the corresponding inpainted view. Notice that the synthesized views maintain consistency with each other; however, view-dependent effects still remain (e.g., the lighting on the uncovered part of the piano). Please see our supplement for additional scenes, and our project website with videos for better visualization.

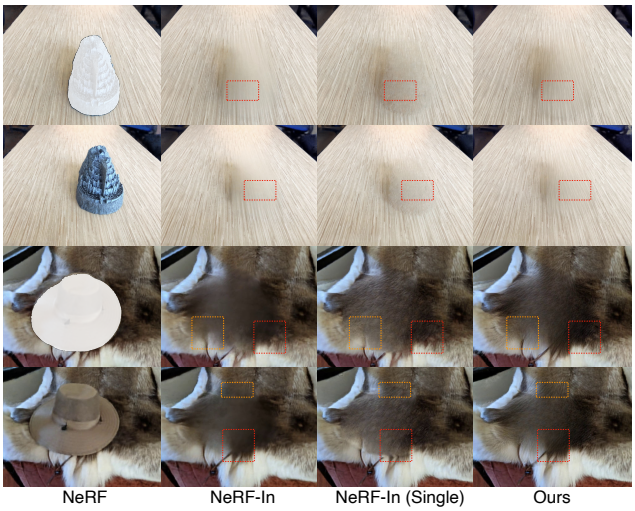

Figure 7. Qualitative comparisons with other baselines. Columns: novel views of the scene, synthesized by a NeRF (on the unmasked images), NeRF-In, NeRF-In (with a single masked training image), and our approach. NeRF-In is significantly more blurry, while NeRF-In (single) tends to have difficulty with details closer to the edge of the mask boundary (zoom into boxes for examples).

Importance of Accurate Masks. Here, we examine the impact of accurate masks on inpainting, via variable dilations of the object masks with a 5×5 kernel. Larger masks lead to relying more on the view-inconsistent outputs and hallucinations of the 2D inpainters, while smaller masks may permit parts of the unwanted object's edges to remain and confuse the 2D inpainter. A subtle dilation is also useful for reducing the effect of shadows. This balance between over-masking and under-masking is demonstrated in Table 4, with five dilation iterations found to be optimal and therefore used for all other experiments.

Table 4. Quantitative evaluation of our 3D inpainting method with different numbers of input views (upper half) and different levels of mask dilation (lower half).

Input Views	**10**	**20**	**40**	**60**
LPIPS↓	0.4726	0.4713	0.4667	**0.4654**
FID↓	171.19	172.63	158.02	**156.64**
Dilation	**45**	**25**	**5**	**0**
LPIPS↓	0.6102	0.5369	**0.4654**	0.4904
FID↓	283.01	230.03	**156.64**	164.66

6. Conclusion

In this paper, we presented a novel approach to inpaint NeRF scenes, which enforces viewpoint consistency based on image and geometric priors, given an object mask for a single input view. In addition, we provided a multiview segmentation method that simplifies the annotation process by

using a set of sparse pixel-level clicks on (and around) the undesired object and translating them into a 3D mask that can be rendered from novel views. We provided extensive experiments to show the effectiveness of our segmentation and inpainting methods. Finally, we introduce a dataset that not only addresses the current lack of challenging benchmarks for multiview inpainting, but which we believe can assist future advances in this new line of research.

Acknowledgement. This work was conducted at Samsung AI Centre Toronto and it was funded by Mitacs and Samsung Research, Samsung Electronics Co.,Ltd.

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. *ICCV*, 2021. 3

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 3

[3] Yash Bhalgat. Hashnerf-pytorch. https://github.com/yashbhalgat/HashNeRF-pytorch/, 2022. 4, 1

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 3, 6, 7, 4, 5

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3

[6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. MobileNeRF: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *arXiv*, 2022. 1, 3

[7] Lu Chi, Borui Jiang, and Yadong Mu. Fast Fourier convolution. In *NeurIPS*, 2020. 2

[8] PaddlePaddle Contributors. PaddleSeg, end-to-end image segmentation kit based on PaddlePaddle. https://github.com/PaddlePaddle/PaddleSeg, 2019. 2

[9] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *CVPR*, 2003. 2

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, June 2022. 3, 1

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2

[12] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2

[13] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017. 2

[14] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. Drop the GAN: In defense of patches nearest neighbors as single image generative models. In *CVPR*, 2022. 5

[15] Yuying Hao, Yi Liu, Zewu Wu, Lin Han, Yizhou Chen, Guowei Chen, Lutao Chu, Shiyu Tang, Zhiliang Yu, Zeyu Chen, and Baohua Lai. Edgeflow: Achieving practical interactive segmentation with edge-guided flow. In *ICCV Workshops*, 2021. 2, 3, 6

[16] James Hays and Alexei A Efros. Scene completion using millions of photographs. In *SIGGRAPH*, 2007. 2

[17] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 3

[18] Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. Escaping Plato's cave: 3D shape from adversarial rendering. In *ICCV*, 2019. 2

[19] 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. *NeurIPS*, 2017. 6

[20] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. EfficientNeRF: Efficient neural radiance fields. In *CVPR*, 2022. 1

[21] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ToG*, 2017. 2

[22] Jitesh Jain, Yuqian Zhou, Ning Yu, and Humphrey Shi. Keys to better image inpainting: Structure and texture go hand in hand. In *WACV*, 2023. 2

[23] Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. A comprehensive review of past and present image inpainting methods. *CVIU*, 203:103147, 2021. 2

[24] Varun Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Kaeser, William T Freeman, David Salesin, Brian Curless, and Ce Liu. Slide: Single image 3D photography with soft layering and depth-aware inpainting. In *ICCV*, 2021. 2

[25] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. CoNeRF: Controllable neural radiance fields. In *CVPR*, 2022. 3

[26] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for editing via feature field distillation. In *NeurIPS*, 2022. 6

[27] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. MAT: Mask-aware transformer for large hole image inpainting. In *CVPR*, 2022. 2

[28] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 3

[29] David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. In *CVPR*, 2022. 3

[30] Hongyu Liu, Bin Jiang, Yi Xiao, and Chao Yang. Coherent semantic attention for image inpainting. In *ICCV*, 2019. 2

[31] Hao-Kang Liu, I-Chao Shen, and Bing-Yu Chen. NeRF-In: Free-form NeRF inpainting with RGB-D priors. In *arXiv*, 2022. 3, 6, 7

[32] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *ICCV*, 2021. 3

[33] Yi Liu, Lutao Chu, Guowei Chen, Zewu Wu, Zeyu Chen, Baohua Lai, and Yuying Hao. PaddleSeg: A high-efficient development toolkit for image segmentation. In *arXiv*, 2021. 2

[34] 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. *ToG*, 2019. 5

[35] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 4, 5, 6, 7

[36] Ashkan Mirzaei, Yash Kant, Jonathan Kelly, and Igor Gilitschenski. LaTeRF: Label and text driven object radiance fields. In *ECCV*, 2022. 2, 3, 4

[37] Thomas Müller. Tiny CUDA neural network framework, 2021. https://github.com/nvlabs/tiny-cuda-nn. 4, 1

[38] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 2022. 3, 4, 1, 2

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 1

[40] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 2

[41] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *arXiv*, 2022. 4

[42] Hiba Ramadan, Chaymae Lachqar, and Hamid Tairi. A survey of recent interactive image segmentation methods. *CVM*, 2020. 2

[43] Chaolin Rao, Huangjie Yu, Haochuan Wan, Jindong Zhou, Yueyang Zheng, Yu Ma, Anpei Chen, Minye Wu, Binzhe Yuan, Pingqiang Zhou, Xin Lou, and Jingyi Yu. Icarus: A specialized architecture for neural radiance fields rendering. In *arXiv*, 2022. 1

[44] Zhongzheng Ren, Aseem Agarwala, Bryan Russell, Alexander G. Schwing, and Oliver Wang. Neural volumetric object selection. In *CVPR*, 2022. 2, 6, 7, 4, 5

[45] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 2012. 6

[46] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *ICCV*, 2019. 5

[47] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3, 4

[48] 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 *WACV*, 2022. 2, 3, 4, 5, 6, 7

[49] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeuRIPS*, 2020. 2

[50] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in neural rendering. In *SIGGRAPH*, 2021. 1, 2

[51] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *3DV*, 2022. 6

[52] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 2

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[54] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. NeSF: Neural semantic fields for generalizable semantic segmentation of 3D scenes. In *TMLR*, 2022. 7

[55] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields. *CVPR*, 2022. 3

[56] 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 *CVPR*, 2021. 3, 5

[57] Wenguan Wang, Tianfei Zhou, Fatih Porikli, David Crandall, and Luc Van Gool. A survey on deep learning technique for video segmentation. In *arXiv*, 2021. 3, 6

[58] Yi Wang, Xin Tao, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. Image inpainting via generative multi-column convolutional neural networks. In *NeurIPS*, 2018. 2

[59] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF–: Neural radiance fields without known camera parameters. In *arXiv*, 2021. 3

[60] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-time

view synthesis with neural basis expansion. In *CVPR*, 2021. 5

[61] Chaohao Xie, Shaohui Liu, Chao Li, Ming-Ming Cheng, Wangmeng Zuo, Xiao Liu, Shilei Wen, and Errui Ding. Image inpainting with learnable bidirectional attention maps. In *ICCV*, 2019. 2

[62] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, and Shiguang Shan. Shift-net: Image inpainting via deep feature rearrangement. In *ECCV*, 2018. 2

[63] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, 2021. 2, 3, 6, 7

[64] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3D-aware scene manipulation via inverse graphics. *NeuRIPS*, 2018. 2

[65] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. NeRF-Supervision: Learning dense object descriptors from neural radiance fields. In *ICRA*, 2022. 5, 1

[66] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 3

[67] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 3

[68] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *CVPR*, 2018. 2

[69] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. NeRF-editing: geometry editing of neural radiance fields. In *CVPR*, 2022. 3

[70] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. ARF: Artistic radiance fields. In *ECCV*, 2022. 5

[71] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 2, 3, 4, 6

[72] 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. In *ICLR*, 2021. 2

[73] Yunhan Zhao, Connelly Barnes, Yuqian Zhou, Eli Shechtman, Sohrab Amirghodsi, and Charless Fowlkes. Geofill: Reference-based image inpainting of scenes with complex geometry. In *arXiv*, 2022. 2

[74] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *CVPR*, 2019. 2

[75] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 2, 3, 6

[76] Shuaifeng Zhi, Edgar Sucar, Andre Mouton, Iain Haughton, Tristan Laidlow, and Andrew J. Davison. iLabel: Interactive neural scene labelling. In *arXiv*, 2021. 2, 3

A. Summary

We provide details of our proposed refinement process in Appendix B. In Appendix C, additional details about our implementation are provided for reproducibility. Appendix D contains further qualitative experiments showing the visual performance of our multiview segmentation and inpainting methods. We also provide a supplementary video and a website with video renderings of the scenes with and without inpainting for better visualization. In Appendix E, we provide an ablation study measuring the impact of additional training stages to segmentation performance. Appendix F provides an overview of all of the scenes in our introduced dataset. For completeness, we provide an extended version of the background on NeRFs in Appendix G. Finally, due to the generative nature of inpainting, we provide an ethics statement in Appendix H.

B. Refinement Details

For pixel values that are only visible in some of the views, we use mask refinement to project them to all of the input views, as introduced in § 4.2.4 in the main paper. This refinement reduces the masked area and leads to better inpaintings due to a decreased need for hallucination. Consider a source image, I_s, its corresponding depth, D_s, and mask, M_s. For each target image, depth, and mask tuple, (I_t, D_t, M_t), and for every masked pixel in the source view, u_s, we consider the ray passing through u_s: $r_{u_s} = o_{u_s} + td_{u_s}$. The same sampling approach used in the original NeRF paper [35] is performed to sample $\{t_i\}_{i=1}^{N}$ on ray r_{u_s}. At the i-th step, the point represented by t_i is projected into the world coordinate system as:

$$X_i = G_s K^{-1} t_i u_s, \tag{10}$$

where G_s is the source camera pose and K is the camera intrinsic matrix. Next, point X_i is unprojected into the target views to determine which pixel in the target view corresponds to u_s [65]:

$$u_{t,i} = \pi(K G_t^{-1} X_i), \tag{11}$$

where G_t is the camera pose of the target view, and π stands for the perspective projection operation. If $u_{t,i}$ is masked in the target view, t_i is ignored and we go to t_{i+1}. If it is not masked, we check if the depth, $D_t(u_{t,i})$, is consistent with the distance of X_i to the target camera. In case of depth inconsistency, again, t_i is discarded and we proceed to t_{i+1}. If the depths are consistent, the RGB color $I_s(u_s)$ is replaced with $I_t(u_{t,i})$ while unmasking u_s in the source view. Note that for refining $D_s(u_s)$, one can not directly use $D_t(u_{t,i})$ because it is the distance to the source camera. The depth $D_t(u_{t,i})$ is first projected to the world coordinates similar to Eq. 10 as:

$$X_{\text{depth}} = G_t K^{-1} D_s(u_{t,i}) u_{t,i}. \tag{12}$$

The distance of X_{depth} to the source camera is then used to replace $D_s(u_s)$.

For each source image, we visit the target images one by one; if a pixel is able to be refined with respect to a new target image, the refinement is performed, and if a previously refined pixel is able to be refined with a point closer to the source camera, the refinement is updated. We iterate our refinement process multiple times, until no pixel is refined. This makes the process independent of the order of the target views.

Figure 8 shows a toy example to visualize the mask refinement process. The unwanted object is the sofa. For a source and target view, a masked pixel in the source view is considered, and a ray is passed through this pixel. The first two sampled points on this ray are still masked when unprojected into the target view since they fall on the sofa in the 3D world. The next sampled point is unprojected to an unmasked pixel on the target view, but the depth is inconsistent since the target camera sees the basketball from that pixel. Finally, the blue cross shows the fourth sampled point, where the depth is consistent, and the green pixel corresponding to the leaves of the tree is used to refine the source image. The distance of the blue cross to the source camera is used to replace the source depth. In practice, a source pixel is refined only if, after the refinement, the new depth is consistent with at least one of the eight neighbouring pixels in the source view. Figure 9 shows an example of an image from one of the scenes in our dataset, before and after refinement. We also provide corresponding masks to show the effect of our refinement process in reducing the masked area. Note that, following our other experiments in the main paper, the mask before refinement is dilated for five iterations, with a 5×5 kernel.

C. Additional Details

In practice, λ_{LPIPS} and λ_{depth} are set to 0.01 and 1, respectively. Our implementation is primarily in PyTorch [39], except for the encoders and MLP implementation, which use Tiny Cuda NN [37] for efficiency. The models are trained on a single Nvidia RTX A6000 GPU. We use the sparse depth supervision in the unmasked regions of the input views, as in DS-NeRF [10], to obtain more accurate scene geometries. Following Instant-NGP [3, 38], the multi-resolution hash encoder used in our NeRF has 16 levels, each returning two features. The base resolution is set to 16. The MLPs have 64-dimensional hidden layers. The first MLP, which calculates the density, σ (and "Objectness logit", s, for multiview segmentation), has two layers, while the color MLP has three layers. The training images used for our quantitative experiments have 567×1008 pixels (after being downsized four times to avoid memory issues), and all are captured by a Samsung Galaxy S20 FE. To calculate the perceptual loss, at each iteration, a random batch of

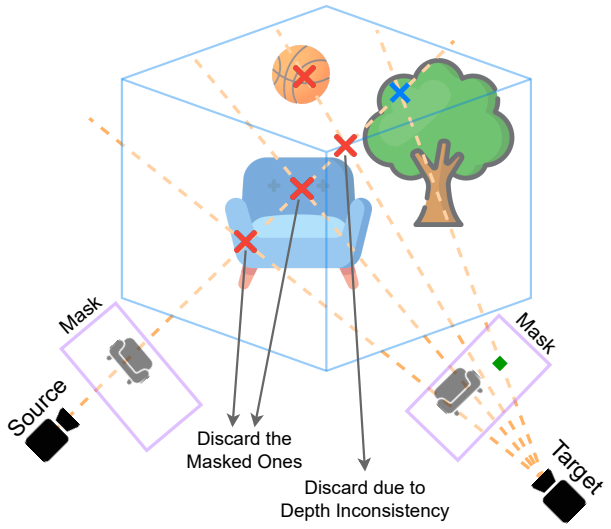

Figure 8. A visualization of our proposed mask refinement. The green pixel depicted in the target view is the final one that is used to transfer color and depth from the target view to the source view. Crosses represent the sampled points, and the blue cross is the final point used for the refinement in this example.

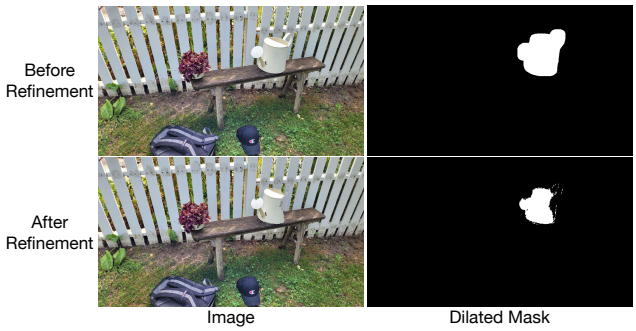

Figure 9. Qualitative example of how refinement can reduce the masked area by substituting pixel values from other views.

four views is selected, and for each of them, a patch is rendered and compared to its inpainted counterpart in the perceptual space. Each patch is 16 times smaller than the original image in each direction, while the stride for sampling the patches is set to two to cover larger areas. This makes the perceptual loss more meaningful, without slowing down the training. As mentioned in the paper, FID and LPIPS are calculated only for the bounding box of the masked region. The mask for test views is rendered using our multiview segmentation model, because the test views do not contain the object and can not be manually masked. Since in the experiments, masks are sometimes dilated, we also expand each side of the bounding box containing the mask in every direction by 10% to make sure that in all of the experiments, the entire hallucinated region is being evaluated.

Table 5. Approximate times that each of the stages in our multiview segmentation and multiview inpainting framework take. These numbers do not include the time spent for human-annotations.

Stage Name	**Time**
Multiview Segmentation	
Interactive Segmentation	<1 second
Video Segmentation	<1 minute
Fitting the Semantic NeRF	$2-5$ minutes
Rendering Training Masks	1 minute
Multiview Inpainting	
Applying the Image Inpainter	<1 minute
Fitting the Inpainted NeRF	$20-40$ minutes

C.1. Approximate Timings

In Table 5, we provide the approximate times that each stage in our framework takes. We use a similar architecture to Instant-NGP [38], which yields fast convergence for our models. Note that the semantic NeRF typically converges to an acceptable geometry even half-way through the fitting iterations, and the remaining iterations are mostly for obtaining a sharp appearance. Since our segmentation and inpainting approaches only use the rendered masks and depths from the semantic NeRF, according to the application, one can trade off quality for speed, and early stop the semantic NeRF to further reduce the segmentation time. For fitting the inpainted NeRF, since we have to render multiple patches and calculate the perceptual loss for each of them, the entire process is slower than the segmentation part. However, according to the fitting times in the literature, this is still a fast NeRF manipulation model for realistic scenes. Note that all of these times can be reduced in the future with faster hardware and underlying models, e.g., better differentiable scene representations.

D. Additional Qualitative Results

Here, we provide additional qualitative examples to show the effectiveness of our multiview segmentation and multiview inpainting methods. Figure 10 is an extended version of Figure 6, and shows four additional qualitative examples of our view-consistent inpainting approach.

Figure 11 shows an example of a single scene being inpainted twice, each time with a different part of the scene being masked. In the upper case, the statue without its concrete base is selected and the base is still in the scene after the inpainting. Notice that parts of the base as well as parts of the ground behind it were not visible in any of the training views. Our model shows consistent plausible hallucination to complete the cylinder shape of the base. The use of the perceptual loss leads to a sharp texture on the grass.

We further provide more qualitative results of our mul-

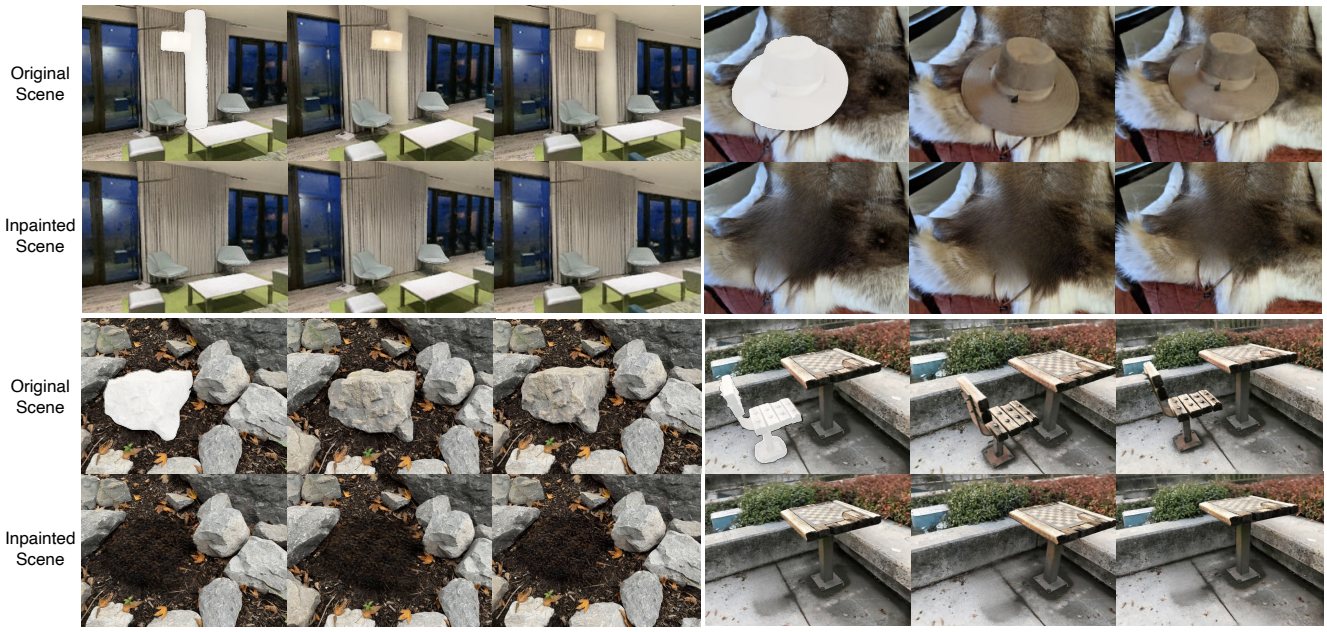

Figure 10. Additional qualitative visualizations of our view-consistent inpainting results, as in Figure 6 in the main paper. Upper rows per inset show NeRF renderings of the original scene from novel views, with the first image also displaying the associated mask. Lower rows show the corresponding inpainted view.

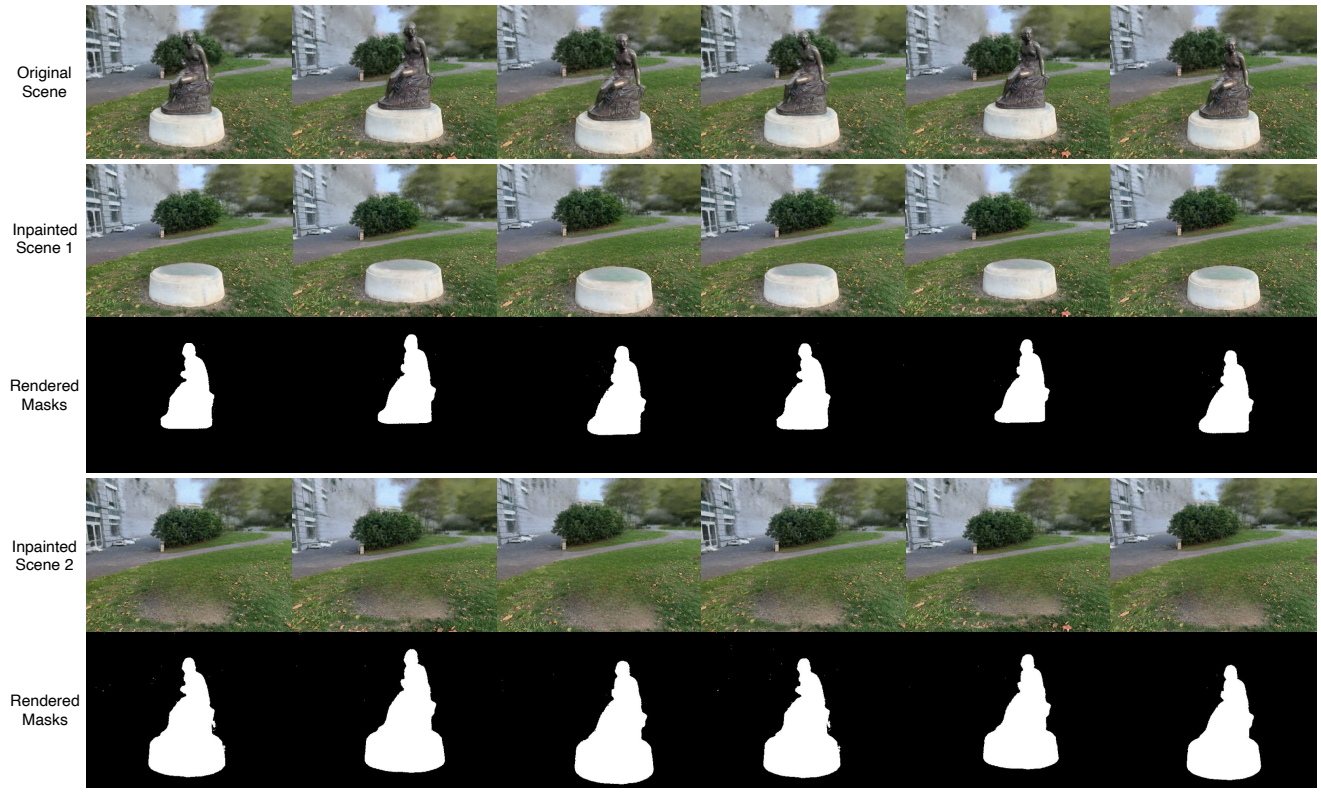

Figure 11. A single scene inpainted with two different masks using our multiview inpainting method.

Table 6. Quantitative evaluation of our proposed multiview segmentation with one, two, and three training stages.

# of Stages	Acc.↑	IoU↑
1	98.85	90.96
2	**98.91**	**91.66**
3	98.89	91.53

tiview segmentation model. Figure 12 is an extension of Figure 5, and shows target views from two scenes, the ground-truth mask in the target views, and the outputs of NVOS [44], video segmentation [4], and our model with or without the two-stage training. As evident in the results, our segmentation model consistently provides coherent masks with sharp accurate edges (zoom into boxes for examples).

E. Multi-Stage Multiview Segmentation

While it has been shown both qualitatively (Figure 5) and quantitatively (Table 1) that our multiview segmentation benefits from our proposed two-stage training, Figure 13 shows that additional training stages do not have a significant effect on the outputs, and thus, two training stages are sufficient. Quantitatively, Table 6 shows that our model with two or three stages of training has similar performance.

F. Our Multiview Inpainting Dataset

Figure 14 contains sample images from our introduced dataset used in our quantitative evaluations. This dataset contains 10 real-world scenes and includes different challenging 3D inpainting segmentation and inpainting scenarios. In the experiments, we use this dataset to provide a quantitative comparison of our inpainting method against the baselines, where our approach outperforms other methods.

G. NeRF: Extended Background

Here, we provide an extended version of the background on Neural Radiance Fields (NeRFs) for completeness. NeRFs [35] encode a 3D scene as a function, $f : (x, d) \rightarrow (c, \sigma)$, that maps a 3D coordinate, x, and a view direction, d, to a color, c, and density, σ. The function f can be modelled in various ways, such as a multilayer perceptron (MLP) with positional encoding [35] or a discrete voxel grid with trilinear interpolation [47], depending on the application and desired properties. For a 3D ray, r, characterized as $r(t) = o + td$, where o denotes the ray's origin, d its direction, and t_n and t_f the near and far bounds, respectively, the expected color is:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)\,\mathrm{d}t, \tag{13}$$

where $T(t) = \exp(-\int_{t_n}^{t} \sigma(r(s))\,\mathrm{d}s)$ is the transmittance. The integral in Eq. 13 is estimated via quadrature by dividing the ray into N sections and sampling t_i from the i-th section:

$$\widehat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))c_i, \tag{14}$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j)$ and $\delta_i = t_{i+1} - t_i$ is the distance between two adjacent sampled points. For simplicity, $c(r(t_i), d)$ and $\sigma(r(t_i))$ are abbreviated as c_i and σ_i, respectively. For the rays passing through pixels of the training views, the ground-truth color, $C_{\text{GT}}(r)$, is available, and the representation is optimized using the reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \sum_{r\in\mathcal{R}} \|\widehat{C}(r) - C_{\text{GT}}(r)\|^2, \tag{15}$$

where $\mathcal{R}$ is a ray batch sampled from the training views.

H. Ethics Statement

There has been a constant debate about 2D generative models and image manipulation techniques, and the concerns regarding potential misuses. The majority of these concerns also apply to the new line of 3D generation and manipulation [41]. In the hands of an adversary, these models can be utilized to manipulate people's perception of reality and generate disinformation. Moreover, the fact that LaMa [48] is used in our implementation results in the inheritance of LaMa's potential undesirable biases in the outputs of our 3D inpainter.

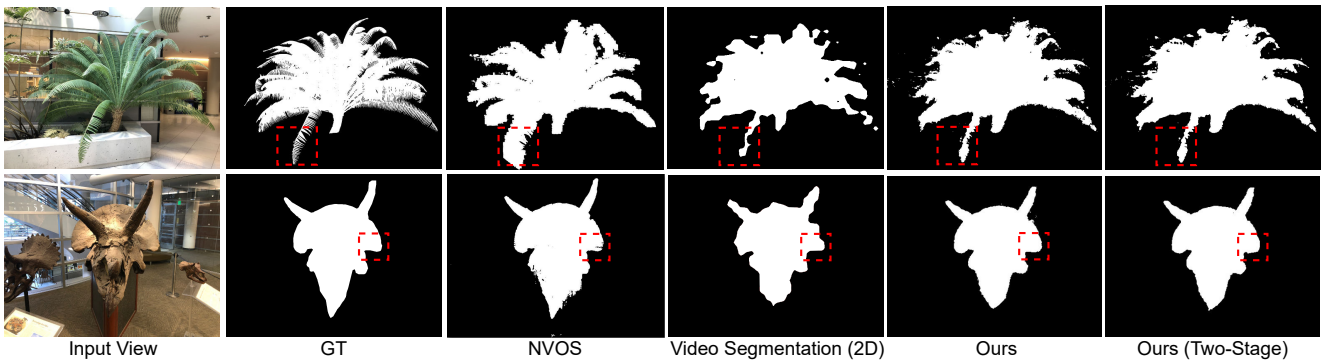

Figure 12. Qualitative comparison, as in Figure 5 in the main paper, of our multiview segmentation model against Neural Volumetric Object Selection (NVOS) [44], Video segmentation [4], and the human-annotated masks (GT).

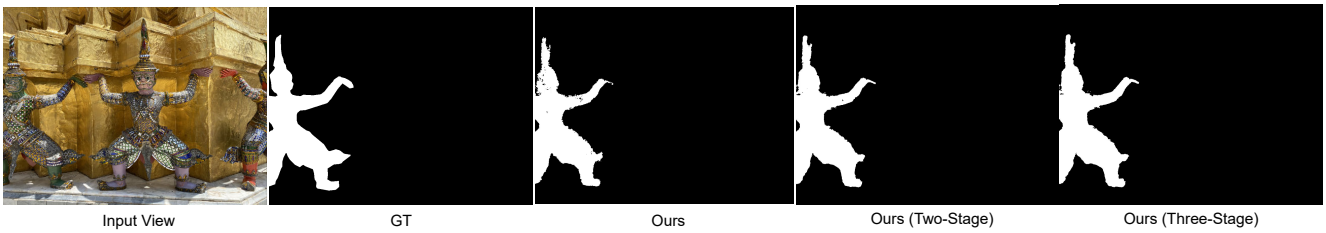

Figure 13. Qualitative comparison of our multiview segmentation model with two-stage and three-stage optimizations. As evident in the results, three-stage optimization does not lead to a significant improvement over the two-stage fitting.

Figure 14. Overview of the 10 different scenes in our introduced dataset for multiview inpainting.