NeRFiller: Completing Scenes via Generative 3D Inpainting
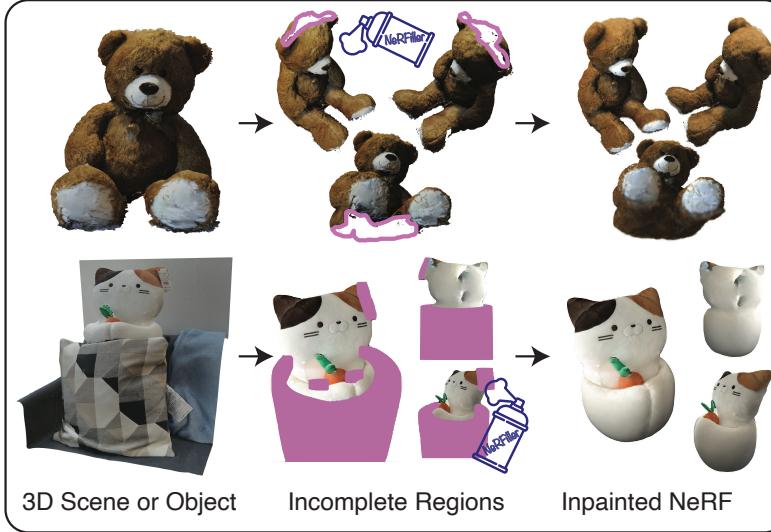
Ethan Weber[1,2] Aleksander Hołyński[1,2] Varun Jampani[1] Saurabh Saxena[1]
Noah Snavely[1] Abhishek Kar[1] Angjoo Kanazawa[2]
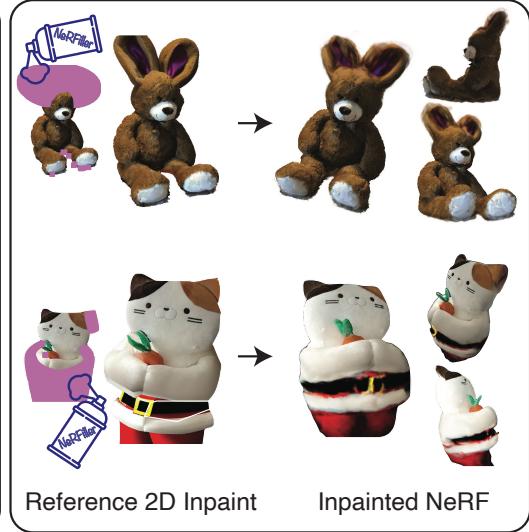[1]Google Research   [2]UC Berkeley

Figure 1. **NeRFiller.** We propose a generative 3D inpainting approach for scene or object completion. Given a 3D capture with incomplete regions (left), our approach completes scenes such as the incomplete teddy bear scan (top left) and deletes unwanted occluders such as the pillow and the price tag (bottom left). We can also control the completions using a reference inpainted exemplar (right) to guide the process.

Abstract

We propose NeRFiller, an approach that completes missing portions of a 3D capture via generative 3D inpainting using off-the-shelf 2D visual generative models. Often parts of a captured 3D scene or object are missing due to mesh reconstruction failures or a lack of observations (e.g., contact regions, such as the bottom of objects, or hard-to-reach areas). We approach this challenging 3D inpainting problem by leveraging a 2D inpainting diffusion model. We identify a surprising behavior of these models, where they generate more 3D consistent inpaints when images form a 2×2 grid, and show how to generalize this behavior to more than four images. We then present an iterative framework to distill these inpainted regions into a single consistent 3D scene. In contrast to related works, we focus on completing scenes rather than deleting foreground objects, and our approach does not require tight 2D object masks or text. We compare our approach to relevant baselines adapted to our setting on a variety of scenes, where NeRFiller creates the most 3D consistent and plausible scene completions. Our project page is at https://ethanweber.me/nerfiller.

1. Introduction

Consider the 3D scanned teddy bear and cat in Figure 1. In many 3D captures such as these, parts of the scene may not be as one desires: there may be unobserved regions such as the bottom of the bear and behind the cat, or there may be unwanted parts such as the price tag on the cat ear. Additionally, one may want to modify a feature, or generate a variety of alternative models, *e.g.*, a bear with bunny ears or a santa cat. All of these tasks require the ability to edit and inpaint content in a 3D-aware and multi-view consistent manner. This is a challenge, since 2D generative inpainting models will not by default generate 3D consistent images. Our goal is to take a step in this direction and present a method that can create new content via scene completion conditioned on a set of multi-view images.

Specifically, we present a 3D scene completion framework called *NeRFiller*, which given a scene and specified parts of the scene to inpaint, returns a 3D scene that is completed in a multi-view consistent manner. Our approach not only completes missing regions (Figure 1, center), but can also generate multiple variations of the missing regions (Figure 1, right). Furthermore, our approach does not require
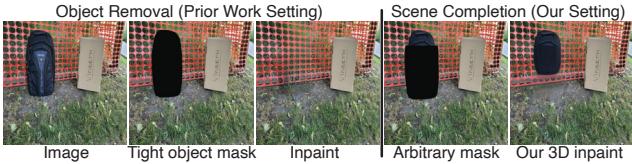
Figure 2. **Object removal vs. scene completion.** We focus on scene completion (right) as opposed to object removal (left). Prior work focuses on removing entire objects with tight masks, while we tackle the more general setting of completing scenes with arbitrary missing regions across wide baselines. More realistic scenarios include missing regions or parts of scenes to edit, as illustrated in Figure 1.

text prompting and can operate from the scene context alone.

We achieve this by proposing a novel approach to generate inpaints with an off-the-shelf 2D generative image model in a manner that encourages multi-view consistency. Specifically, we identify a useful phenomenon in text-to-image diffusion models that we refer to as a *Grid Prior*: denoising four images with missing observations that are tiled in a 2×2 grid results in more consistent multi-view inpaints than inpainting them independently, shown in Figure 3. We propose a method called *Joint Multi-View Inpainting* that generalizes this behavior to more than four images. While this technique results in more 3D consistent inpaints, it is still a 2D-based approach and 3D consistency is not guaranteed. Therefore, we propose a way to distill these inpaints in a global 3D scene representation in an iterative manner.

While there has been a surge of recent works that generate 3D scenes completely from scratch using text [14, 23] or image guidance [28, 29], our approach differs in that we focus on completing scenes given the context of an existing 3D scene. Our approach is related to recent methods that remove a specified object from a scene [36], but we can generate new content that goes beyond completing a textured background, as illustrated in Figure 2. We also do not assume a tight object mask, and can generate a diverse set of inpaints.

To demonstrate the efficacy of our approach, we experiment with a diverse set of scenes including 3D indoor photogrammetry captures lacking coverage in certain areas, 3D scenes with specified missing regions, and 3D objects. While our problem is challenging, we show that NeRFiller can recover more 3D consistent and plausible results compared to recent state-of-the-art methods adapted to our setting.

2. Related Work

Our goal is to complete missing parts of an existing 3D scene. There are several ways to approach this, via 2D inpainting or via distilling a 2D generative model for 3D generation.

2D inpainting. 2D inpainting methods take an image and mask and complete the missing content at the mask location. Early methods relied on inpainting by copying texture from
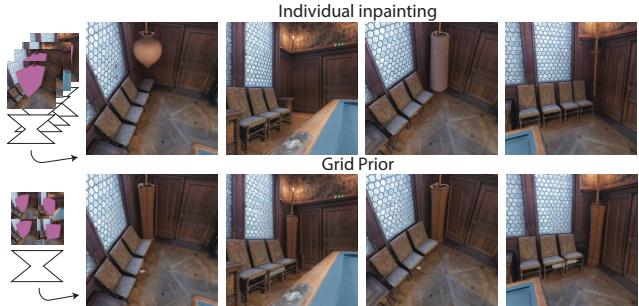

Figure 3. **Grid Prior.** Here we inpaint the corner of the room (left illustrated in pink) with individual inpainting (top) and our Grid Prior method (bottom). Individual inpaints are diverse, while the Grid Prior encourages multi-view consistency.

known regions into the unknown regions [13]. A state-of-the-art model is LaMa (Large Mask inpainting) [47], which is particularly good at infilling large missing areas. It uses fast Fourier convolutions, a large receptive field, and large training masks. This model is highly effective at completing plausible "background" textures within a specified mask (Fig. 2 left) but lacks diverse outputs as it is deterministic. Probabilistic diffusion models [22, 38] have recently produced remarkable results for image generation. They can also be used for inpainting and can generate diverse inpainted outputs. Pixel-based diffusion models do not have to be trained explicitly for inpainting, but can be modified at test-time by setting known regions before each denoising iteration [31]. Latent diffusion models (LDMs) [43] are also effective at inpainting and are efficient because they operate in latent space. However, they require fine-tuning for inpainting with image and mask conditioning. 2D inpainting models can be prompted [1] and/or fine-tuned [49] enabling additional flexibility for downstream applications.

3D generation. 3D generation takes as input text or images and outputs 3D content. The Infinite Nature line of work [26, 28, 29, 41, 59], takes as input a single image and generates immersive fly-through content using a 2D inpainting model queried in an autoregressive manner [28, 29]. Cai et. al. [6] follow this path with a diffusion model, however, none of these approaches can recover a global 3D scene representation. Persistent Nature [7] and related work [9, 12] maintain a latent scene but are completely generative and not conditioned on input image sets. SceneScape [14] and Text2Room [23] use text prompts and 2D inpainters to create a 3D mesh by using an inpainter and depth predictor to successively stitch a mesh. These approaches cannot fix a mistake in the scene if a bad inpaint is made during the successive stitching because no global optimization is performed.

Other methods create 3D content via a global optimization strategy. DreamFusion [39] and related works [55, 62] use the NeRF framework to optimize a 3D volume given
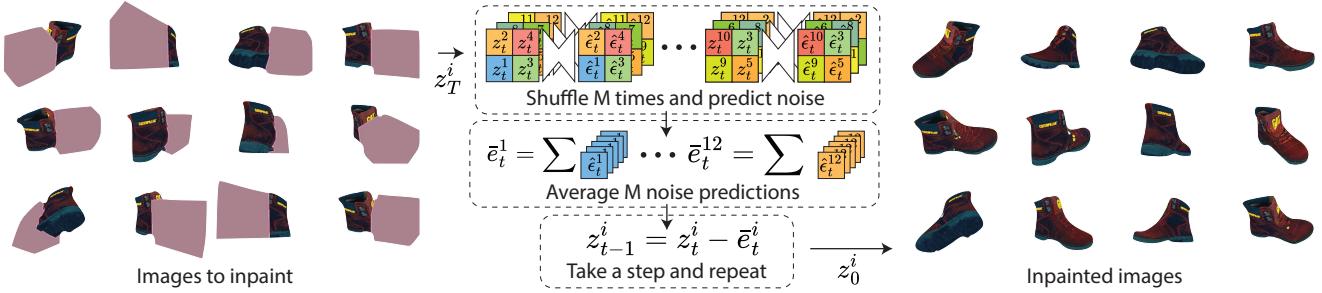
Figure 4. **Joint Multi-View Inpainting.** We enable properties of the Grid Prior with more than four images by averaging diffusion model predictions. We take N images (left), create $N/4$ grids, and obtain a noise prediction from SD [43]. We do this M times and average the noise predictions before taking a denoising step. At z_0, the images (right) are fairly consistent and can be used to train a NeRF with our Inpaint DU method.

a text prompt. Others train models to have 3D consistent properties [27, 30, 50, 57, 62]. Follow-up works leverage 2D diffusion models techniques [15, 44] to create 3D content conditioned on real images [8, 32, 40]. These approaches are not designed for the inpainting task.

3D inpainting. Unlike most 3D generation methods, we ground our inpaints with an actual 3D scene or object that has missing regions (Figure 1). Casual capture [5, 19, 20, 33, 51] or NeRFs [25, 34] is a use-case as they often contain artifacts when rendered from novel views [17, 56]. Most relevant to scene the completion setting is the object removal setting (see Figure 2). These works remove foreground objects from NeRF captures [36, 52, 58]. They do this by inpainting each image in a NeRF dataset once and training with various losses including patch-based perceptual losses and depth regularization. [35] enables inpainting from a reference image. A variety of these methods are evaluated on the SPIn-NeRF dataset, which is in the forward-facing LLFF [33] format and has small parallax. Our work uses datasets with a significantly larger baseline.

Our focus is on the more general scene completion setting, which is related to editing. IN2N [18] edits a scene using InstructPix2Pix [4], but it cannot hallucinate new geometry. The video editing literature is also relevant, with techniques such as extended attention from Tune-A-Video [16, 60] to encourage consistency in edited video frames. However, editing 2D images does not guarantee consistency when lifted to 3D. In our method, we encourage 2D inpaints to converge via iterative NeRF optimization and dataset updates. Moreover, we achieve this with *off-the-shelf* 2D generative models without the need for expensive purpose-trained diffusion models or model fine-tuning.

3. Preliminaries

3.1. Neural Radiance Fields (NeRFs)

Neural radiance fields (NeRFs) [34] represent the 3D geometry and radiance of a scene with neural networks. NeRFs take as input an 3D position (x, y, z) and a viewing direction (θ, ϕ), and output a color and density (c, σ). To train a NeRF f_Θ, a set of calibrated, posed images are used to construct a set of 3D rays $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ for each pixel with known color $C(\mathbf{r})$. During training, these rays are sampled and rendered via volumetric rendering to obtain a color estimate $\hat{C}(\mathbf{r})$. Rays are sampled from training images and the field is optimized with photometric losses $\mathcal{L}_{nerf}(C(\mathbf{r}), \hat{C}(\mathbf{r}))$, e.g., MSE or LPIPS [61]. During inference, a full image is rendered with all rays of the desired camera.

3.2. 2D Diffusion Models

Diffusion models consist of two processes: a forward process q that gradually adds noise to a data sample $z_0 \sim p_{data}(z)$, and a learned reverse process to iteratively denoise a pure Gaussian noise sample $z_T \sim \mathcal{N}(0, 1)$ into a clean image z_0. An intermediate noisy z_t can be obtained from the clean image by adding noise ϵ with scaling $\bar{\alpha}_t$, where $z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$. The diffusion model ϵ_θ predicts noise $\hat{\epsilon}$ present in the image z_t as $\hat{\epsilon} = \epsilon_\phi(z_t, t, c)$. t is a time indicating how much noise is in the sample, and c is a general form of conditioning (e.g., images, masks, or text). During training, random noise ϵ and t are sampled and the objective $\mathcal{L}_{diff} = ||\hat{\epsilon} - \epsilon||^2$ is minimized. With the prediction $\hat{\epsilon}$ at time t, a reduced noise z_{t-1} can be obtained by $z_{t-1} = z_t - \hat{\epsilon}$ (where we omit the scaling of $\hat{\epsilon}$ for simplicity). Repeating this until z_0 yields a fully denoised sample. Stochastically training with c (conditionally) and without c (unconditionally) enables classifier-free guidance (CFG) [21] during inference time. In practice, z is latents since we are using SD (Stable Diffusion) [43], but in general we can map from z to higher resolution pixels x with an encoder $\mathcal{E}(x)$ and decoder $\mathcal{D}(z)$.

Using a diffusion model as a prior. Diffusion models have advantages over other models (e.g., GANs and deterministic inpainters [47]) because they can be used a a prior to optimize underlying variables such as the parameters of a 3D NeRF f_Θ with methods like score distillation sampling (SDS) [39, 54]. When used as a prior for NeRFs, the objective is to find the best Θ such that a rendered image x
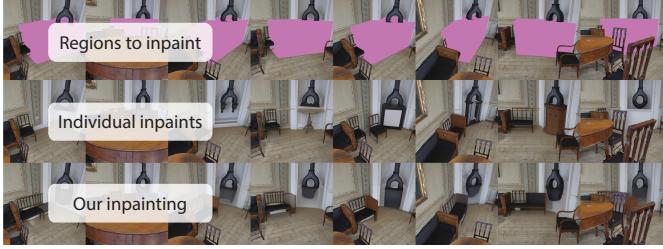
Figure 5. **Joint Multi-View Inpainting Examples.** The top images are inpainted with SD [43] without any text conditioning (middle) and with our Joint Multi-View Inpainting method (bottom). Our joint inpaints are more multi-view consistent.

has high likelihood under the diffusion model prediction ϵ_ϕ. SDS involves rendering an image, adding partial noise, and updating the NeRF such that the diffusion model can predict the added noise. IN2N [18] introduced a variant of this method coined Dataset Update (DU). Instead of backpropping based on the diffusion model prediction, DU renders an image, adds partial noise, and takes multiple steps to recover an estimated clean image x_0. The clean image is added to the dataset and used to supervise the NeRF. Every S iteration, another image is replaced. The DU supervision signal will be slightly delayed since images are cached for several iterations, while SDS provides immediate gradients corresponding to the current render. However, we use the DU method in our work because it has a few advantages over SDS in terms of implementation: We can obtain higher-resolution supervision (albeit slightly delayed) with less GPU memory and we can update a large batch of images simultaneously (e.g., 40 images). We find that large batch updates are important for our inpainting task since we are changing the NeRF geometry, unlike prior works that focus purely on modifying appearance [18, 37].

4. Method

Our method, NeRFiller, aims to complete a missing region within a 3D scene by using an inpainting prior from a generative 2D diffusion model. This problem statement poses a number of challenges. First, the inpainted estimates from a 2D diffusion model are diverse, and may vary from sample to sample. This requires a consolidation mechanism to ensure that the completed 3D scene contains one salient inpainted result, as opposed to the average of all possibilities. Second, 2D inpainting models are not trained for 3D consistency, and will therefore provide estimates that cannot be explained by a single 3D scene, even if they correspond to the same approximate style or content. In the following, we describe our approaches to tackle these problems. In Section 4.1, we describe how we encourage the inpainted outputs from a diffusion sampling process to be 3D consistent. In Section 4.2, we describe an iterative 3D scene optimization method that uses these inpainted images to optimize for a globally consistent inpainted 3D scene. NeRFiller builds on an observation that inpainting a grid of images encourages the outputs to
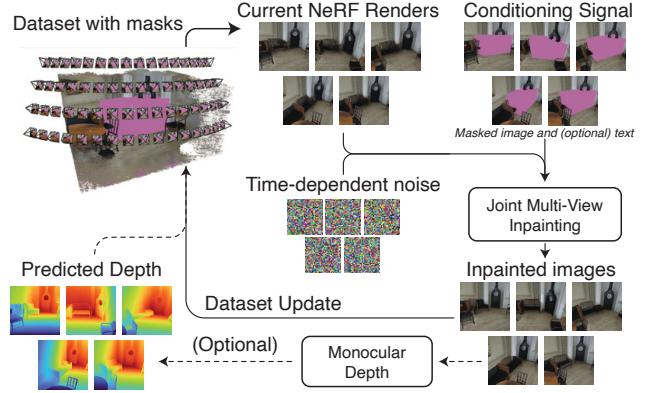

Figure 6. **Inpaint Dataset Update.** Every S iterations, we update the unknown pixels of the NeRF training images. We render N images, add partial noise, and jointly inpaint with a conditioning signal. We (optionally) predict the depth and update the dataset.

have similar appearance, and we extend this idea to an arbitrarily large collection of images through a joint sampling approach.

4.1. Multi-view consistent inpainting

A core challenge in 3D inpainting with a 2D generative model is getting the outputs of the 2D model to be consistent across views. This challenge stems from the multimodality of the output distribution: in most cases, there are many plausible inpaintings, and sampling multiple consistent images remains an open research problem.

Grid Prior. While inpainting multiple viewpoints independently may produce inconsistent results, one interesting discovery is that consistency can be achieved by tiling the input images into a grid and treating the *grid* of images (and their corresponding masks) as a single inpainting target. This grid-based prior can produce more 3D consistent views, both in coarse appearance and approximate scene structure (illustrated in Figure 3). We hypothesize that this phenomenon results from similarly structured examples in Stable Diffusion's training dataset: sets of observations depicting the same scene or object organized as a grid (e.g., screenshots of online product photos). Similar properties were also explored in visual-prompting [1].

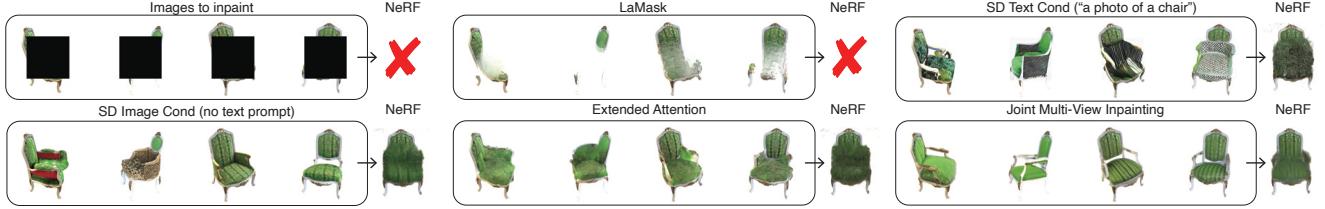More specifically, in order to inpaint four images consis-

Figure 7. Inpainting methods. Inpainting methods produce inconsistent inpaints. We show various inpainting methods (boxed) and use a collection of them to train a NeRF. A resulting render is shown on the right of each method. Using our Grid Prior and Joint Multi-View Inpainting creates reasonably consistent inpaints and a plausible NeRF. $\mathcal{X}$ means the NeRF failed and resulted in white everywhere.

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	7.76	0.71	0.37
LaMask	19.58	0.89	0.20
SD Text Cond	12.56	0.73	0.32
SD Image Cond	14.15	0.76	0.28
Extended Attention	14.57	0.77	0.27
Grid Prior	14.43	0.80	0.25
Joint Multi-View Inpainting	15.89	0.82	0.23

Table 1. Multi-view consistent inpainting. We inpaint images and train a NeRF for the 8 scenes of the NeRF synthetic dataset. Better metrics indicate more consistency of the NeRF 3D reconstruction with the 2D inpaints. Note that LaMask achieves the best results as it often copies the white background into the hole (see Figure 7) and results in a failed NeRF that is totally white.

tently, one can downsample them and their corresponding inpainting masks to quarter-resolution and tile them as a 2×2 grid. This grid is fed through the 2D inpainting model (as a single image would) to get as output four inpainted images with consistent content. More formally, let $\mathcal{G}$ be the downsampling and grid operation for four images and let $\mathcal{G}^{-1}$ undo this. We can grid four images latents $z_t^1, z_t^2, z_t^3, z_t^4$ as follows:

$$\{\hat{\epsilon}_t^1, \hat{\epsilon}_t^2, \hat{\epsilon}_t^3, \hat{\epsilon}_t^4\} = \mathcal{G}^{-1}(\epsilon_\phi(\mathcal{G}(\{z_t^1, z_t^2, z_t^3, z_t^4\}))) \quad (1)$$

and take a denoising step with $z_{t-1}^i = z_t^i - \hat{\epsilon}_t^i$. In principle, this approach is similar to recent methods that use *extended attention*, i.e., shared keys and values in the attention operations across a set of parallel sampling processes [60]. Our approach does not share attention features but instead shares context with other images via the diffusion U-Net receptive field that sees 4 tiled images at a time. In our experiments, we compare to *extended attention* and demonstrate that our grid prior more effectively inpaints 3D consistent content when used with our Joint Multi-View Inpainting method.

Joint Multi-View Inpainting. While effective at inpainting a set of images consistently, applying the grid prior to a larger set of images poses additional challenges. Increasing the number of images in the grid proportionally decreases the output resolution of each image (e.g., arranging a set of 2×2 images in a grid reduces each image's resolution by 4, a 3×3 grid by 9, and so on). For the purpose of producing high-quality inpainting results, we would like to minimize

any loss in image detail. Therefore, we propose a method that uses the above grid prior in a joint sampling process, inspired by MultiDiffusion [2]. In each sampling step, we shuffle all the input images into a set of 2×2 grids. We repeat this M times and before taking a sampling step, the score estimate for each image is combined across all the grid combinations in which it was seen. This causes the inpainting estimates to be gradually shared across the entire dataset, effectively increasing the grid size without further reducing effective resolution.

Figure 4 describes this procedure. More formally, for a batch of N images, we randomly permute their order, construct $N/4$ grids, and predict the noise for M iterations ($j \in [1, M]$), as follows:

$$\{\hat{\epsilon}_t^{1j} \ldots \hat{\epsilon}_t^{Nj}\} = \mathcal{G}^{-1}(\epsilon_\phi(\mathcal{G}(\{z_t^{1j} \ldots z_t^{Nj}\}))) \quad (2)$$

and step with $z_{t-1}^i = z_t^i - \sum_{j \in M} \hat{\epsilon}_t^{ij}$ from z_T to z_0. Qualitative results are shown in Figure 5.

4.2. Completing 3D Scenes

The proposed Joint Multi-View Inpainting enables inpainting images in a more 3D consistent manner than other inpainting methods. Next, we describe how to distill these 2D inpainting results into a single 3D reconstruction. We refer to our method as Inpaint Iterative Dataset Update (Fig. 6), or **Inpaint DU**, as it derives from IN2N's [18] Iterative DU method (see Sec. 3.2). In contrast to IN2N, which begins with a complete NeRF reconstruction and uses a model conditioned on complete 2D observations, our task requires us to train a complete NeRF from images with masked unknown regions. As in IN2N, we begin training with a dataset of original (known) pixels, and update the dataset over training by adding or replacing the set of initially unknown pixels with the inpainted estimates. Fig. 6 illustrates this procedure. Specifically, every S steps, we render the set of N training views, encode them into latents z_0^i and then partially noise them before feeding them to SD. We sample from these partially noised inputs using the proposed Joint Multi-View Inpainting strategy, then use the resulting images to replace the corresponding images in the dataset. This process is both prefixed and suffixed by an encode and decode operation, since the base model is a latent diffusion
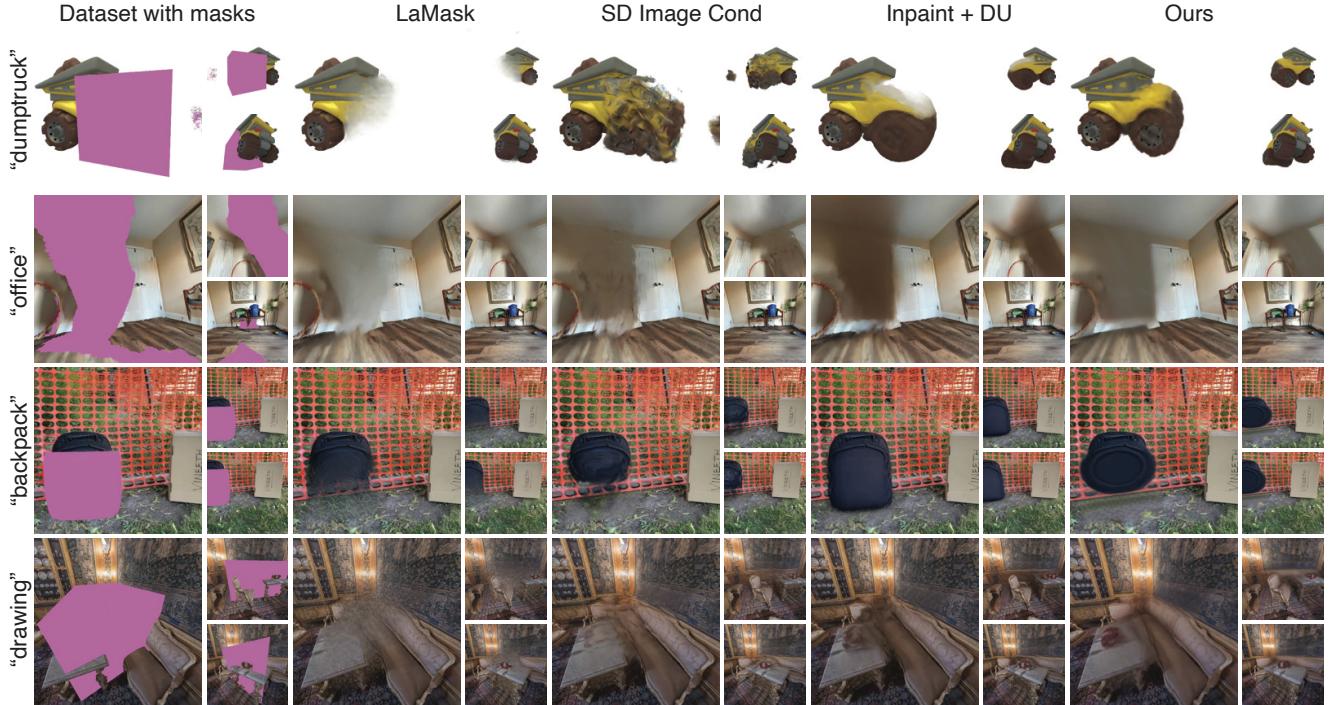
Figure 8. **Qualitative NeRF results.** On the left, we show various scenes with pink regions to be completed. We compare NeRFiller (far right) against baselines adapted to our scene completion setting. The "office" scene is missing parts of the wall, floor, and under the chairs.

model. We repeat this process many times while linearly annealing t from full noise $t = 1$ to $t = t_{\min}$. In practice, we set $t_{\min} = 0.4$, since we find that low noise values result in quality degradation. We observe that over the course of optimization, our inpainted images become gradually more consistent (Fig. 9) as the added geometry and texture begin to take form. Annealing t helps encourage the inpainting to converge to a single result rather than making large changes late in training.

Depth regularization. Inpaint DU optionally incorporates depth supervision to improve inpainted scene geometry. After each dataset update, we predict the depth for all images with ZoeDepth [3]. We use a relative depth ranking loss [53] in the inpainted regions (but not on the known pixels). We use a ranking loss because it's a softer constraint than metric depth supervision, where errors in scale-and-shift alignment could more easily harm the 3D scene geometry. *We only apply depth supervision for our main method to indoor scenes and not objects, since we empirically noticed that [3] performs less consistently when the background is a solid color (e.g., white or black).*

5. Experiments

We compare NeRFiller for 3D scene completion to various inpainting baselines. We first investigate various inpainting strategies on multi-view synthetic scenes to gauge how effective a deterministic inpainter [47] is compared to

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	14.71	0.78	0.26	3.71	675
LaMask	27.39	0.90	0.05	3.76	643
SD Image Cond	22.03	0.86	0.11	3.68	665
Inpaint + DU	26.60	0.89	0.08	3.76	660
Ours w/o depth	28.41	0.92	0.06	3.72	682
Ours	28.28	0.91	0.06	3.73	696

Table 2. **Quantitative NeRF results.** We report various metrics averaged over our 10 scenes to quantify consistency of the 3D NeRF with the dataset inpaints (left), as well as novel-view metrics (right), such as "Corrs" (number of high-quality correspondences between random pairs of frames) for geometry.

SD [43], sampled in various ways. After establishing that our *Joint Multi-View Inpainting* demonstrates multi-view inpainting properties, we evaluate our full method on 10 scans with missing regions. We compare NeRFiller to various object-removal baselines, *adapted to our setting*, to complete missing regions. Finally, we analyze the parameters of our method and show an application to reference-guided scene completion. We conduct experiments with Nerfstudio [48] and provide implementation specifics in the appendix.

5.1. 3D consistent image inpainting

Our goal is to evaluate various 2D inpainting models and strategies to quantify their 3D consistency.

Setting and evaluation. For these experiments, we take the testing split of the NeRF synthetic dataset [34] (8 scenes of
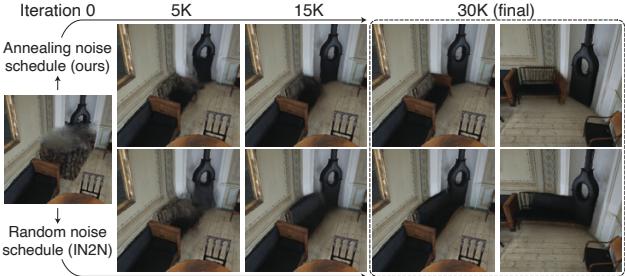
Figure 9. **Noise schedule.** We anneal the amount of noise we add to the NeRF renders when making a dataset update, while I-N2N chooses random noise each time. Annealing the noise produces sharper results (top) while I-N2N's noise schedule introduces significant blur (bottom).

200 images each). We resize each image to 512×512 resolution and mask out the center of each image with a 256×256 region to inpaint (see Figure 7 top left). We inpaint all images with various methods and train a Nerfacto NeRF model [48] on 180 equally spaced images and evaluate metrics on the remaining 20 images. We use the standard NeRF metrics because they capture how similar the 3D reconstruction is to the 20 hold-out evaluation images. When rendering for evaluation (right of inpainted images in Figure 7), we push the near plane slightly forward to avoid including any floaters hiding in front of the cameras.

Baselines. Our baselines are the following:
- *Masked NeRF* - No inpainting, only train on known pixels.
- *LaMask* - LaMa [47] inpainting model.
- *SD Text Cond* - SD using a text CFG with prompt "a photo of {description}". See the appendix for text prompts.
- *SD Image Cond* - SD with only image CFG.
- *Extended Attention* - SD with only image CFG and extended attention [60].

For *Extended Attention* and *Grid Prior*, we inpaint in batches of 5 and 4, respectively, and for *Joint Multi-View Inpainting*, we inpaint 40 images simultaneously with $M = 8$ diffusion averaging steps. To inpaint additional batches of 40 images, we set 20 in the batch as known. This enables fitting within the memory constraints of a 16 GB GPU.

Results. Our results are shown in Tab. 1, where we see that *Joint Multi-View Inpainting* achieves the best metrics in multi-view consistent inpainting. Our results show that we have achieved some level of multi-view consistency. Our images look the most consistent in Figure 7 (bottom right), and the trained NeRF looks plausible. The other methods yield significant blur in the NeRF reconstruction. We provide videos on the project page showing the NeRF results for each method.

5.2. Completing large unknown 3D regions

In this setting, our goal is to complete missing regions in 3D content. We construct a set of 10 datasets consisting
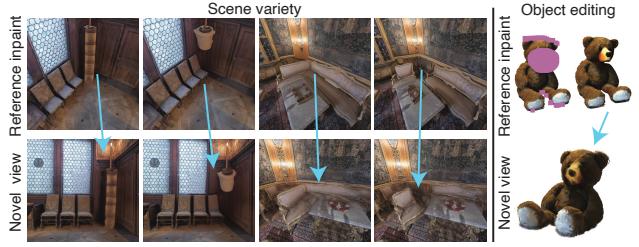

Figure 10. **Reference-based completion.** Given a reference inpaint (top row), we propagate it into a 3D NeRF (bottom row).

of various 3D content. For some scenes e.g., the backpack from [36], we modify their provided mask to include part of the object (Fig. 2) to convert it to the scene completion setting. For other scenes, we simply want to fill in any missing details (e.g., parts of walls). Some of the meshes are missing vertices after multi-view stereo reconstruction, e.g., "bear" and "office". For others, we place a large 3D occluder in the scene to simulate the scene completion setting. The pink regions in Fig. 8 (left) shows the areas to complete. We create the datasets by rendering ∼60 novel views looking at the occluded region. Importantly, the rendered images have enough known pixels to provide context to the inpainting model that the images observe the same scene from different camera viewpoints. Our datasets have much more parallax than the forward-facing scenes of [36]. Our appendix provides details on our data, including where we obtained our 3D content, mostly from Objaverse [10, 11] and Sketchfab.

Evaluation. The evaluation is similar to Sec. 5.1 for the dataset images. However, in this case our task is to construct a scene for good novel-view synthesis, so we use all images for both training and evaluation. We compute NeRF metrics on the entire images, where we compare the final rendered images with the latest version of the inpainted region. For methods that inpaint once without DU (e.g., *LaMask*), we compare against the first and only inpaints. For DU methods, we compare against the latest round of inpaints. *Our metrics are against inpainted images which serves to evaluate the consistency of the scene because there is no ground-truth solution.* We also report novel-view metrics, computed on a custom 10 second 30 FPS camera path novel views that moves around the scene. We also report an image quality metric MUSIQ [24] and a geometry metric. For geometry, we report the number of high-quality LoFTR [45] correspondences between 100 randomly sampled pairs of frames. More high-quality matches should correlate with better multi-view consistency and fewer extreme view-dependent effects that destroy realism. Please see the Appendix for more details.

Baselines. We implemented the following baselines:
- *Masked NeRF* - no inpainting, where we train a Nerfacto model [48] only in the known pixel locations,
- *LaMask* - Inpaint once with LaMa [47] and train with

patch-based perceptual losses. This is our adaptation of SPIn-NeRF [36].

- *SD Image Cond* - Inpaint once with SD. This is similar to InpaintNeRF360 [52] but without text since we find in Sec. 5.1 that text CFG produces very inconsistent inpaints.
- *Inpaint + DU* - An adaption of IN2N [18] for our setting, which inpaints one image at a time with the SD inpainting model and our annealed noise schedule.

Results. Some qualitative results are shown in Fig. 8 and full videos for all 10 scenes are provided in the appendix. *LaMask* and *SD Image Cond* are both inpaint-once methods and therefore create large blurry regions in the NeRF, but between the two, *LaMask* is smoother since its deterministic inpainter [47] is less creative than SD in its outputs. LaMa [47] tends to copy background textures into the mask region. From certain views, *Inpaint + DU* looks sharp due to inpainting individually at full resolution; however, it has geometric inconsistencies and view-dependent effects which are crisp from some angles and blurry in others. Our method looks the most consistent with plausible outputs, although its ability to create consistent high-frequency texture details may be improved. We provide quantitative results showing that our final renders are most similar with the latest round of inpaints (Tab. 2 left). For novel-view metrics, we obtain the most correspondences (Tab. 2 right). Note that although we make an effort to capture the results quantitatively, there is no singular ground truth and therefore the results are best discerned qualitatively by viewing videos.

5.3. Reference-based inpainting

In some situations it is desirable to have control over the content used to complete the scene. NeRFiller can be easily adapted to 3D inpaint with respect to a user-provided reference inpaint. To do this, we first inpaint from one view and use it to prompt our Grid Prior update method. We ensure that each grid has the one reference inpaint when passed through SD. This ensures that all U-Net predictions are influenced by the reference inpaint so that new inpaints are more likely to be consistent with the reference. For this, we edit 30 images at a time (instead of 40) and make 10 grids, each with exactly 1 reference inpaint. See Figure 1 and Figure 10 for examples.

5.4. Parameter choices

Noise schedule. It is important to anneal the amount of noise added the rendered images. We start by adding full noise (1.0) and decrease it to 0.4 over the 30K iterations of training. IN2N [18], in contrast, uses a random schedule of choosing between 0.98 and 0.02 each update. This likely works because the InstructPix2Pix model is image conditioned and geometry of the NeRF does not change much. In our case, we are changing the geometry and using their schedule leads to blurry results, shown in Figure 9.
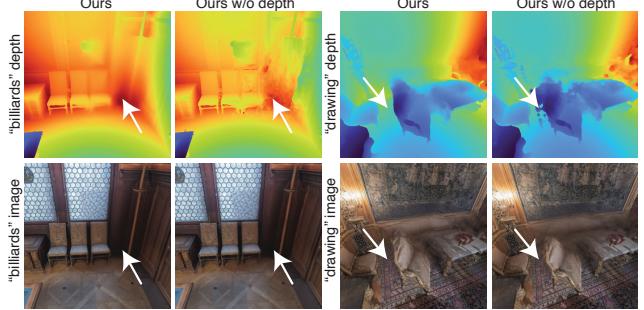

Figure 11. **Relative depth supervision.** Relative depth supervision cleans up geometry (top) without affecting visual quality (bottom).

Depth regularization. We find that adding depth ranking supervision [53] in *Ours* improves geometry but it hardly changes the quantitative or visual results. Our method without depth supervision is still favorable compared to the baselines. In Figure 11, we see that the geometry is significantly improved but the RGB NeRF renderings are nearly indistinguishable. Consequently, we use depth supervision on indoor scenes since having better geometry is favorable for downstream applications such as mesh export.

6. Limitations

Low resolution and blur. Our method recovers coarse geometry quite well but struggles to recover high-resolution detail in regions far way from the training cameras. We suspect this is because our *Joint Multi-View Inpainting* method downsamples images to construct the 2×2 grids. Perhaps a post-processing method to fine-tune SD [46] or a GAN-eRF [42] like optimization could improve the fidelity, but recovering high-frequency details remains a fundamental issue for 3D generative methods, e.g., DreamFusion [39].

Inpainting NeRF casual captures. An application of our approach would be to inpaint deleted content from Nerf-busters [56] or Bayes' Rays [17]. However, these masked-out regions are large (limiting scene context to an inpainter) and furthermore, their mask patterns cause SD [43] to fail without multiple iterations of dilation, as pointed out in Text2Room [23] and in our appendix. One could retrain SD with these mask distributions, but this is out of scope of our method which uses an *off-the-shelf* model.

7. Conclusion

In this paper, we propose a generative 3D inpainting method called NeRFiller, which leverages an *off-the-shelf* 2D inpainting model [43] to complete missing parts of 3D scenes and objects. We discover a unique property of these models where tiling four images into a 2×2 grid produces more consistent inpaints than inpainting them independently. We exploit this property and propose *Joint Multi-View Inpainting*, which enables inpainting many images simultaneously with

more consistency by averaging noise predictions. We show how to use it in the NeRF setting by performing iterative dataset updates. We evaluate against relevant state-of-the-art baselines adapted to our problem setting on a variety of 3D captures. Our approach also enables users to specify how to fill in the missing regions. Many 3D captures are incomplete with holes, and our work presents a framework for completing these missing regions.

Acknowledgements

References

[1] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. Visual prompting via image inpainting. In *ANeurIPS*, 2022. 2, 4

[2] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. In *ICML*, 2023. 5

[3] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. In *arXiv preprint arXiv:2302.12288*, 2023. 6

[4] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 3, 11

[5] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. In *SIGGRAPH*, 2020. 3

[6] Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein. Diffdreamer: Consistent single-view perpetual view generation with conditional diffusion models. In *ICCV*, 2023. 2

[7] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely. Persistent nature: A generative model of unbounded 3d worlds. In *CVPR*, 2023. 2

[8] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3d-aware diffusion models. In *arXiv*, 2023. 3

[9] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Scenedreamer: Unbounded 3d scene generation from 2d image collections. 2023. 2

[10] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. In *arXiv*, 2023. 7

[11] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023. 7

[12] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 2

[13] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 2

[14] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. In *arXiv*, 2023. 2

[15] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *arXiv*, 2022. 3

[16] Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. In *arXiv*, 2023. 3

[17] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' Rays: Uncertainty quantification in neural radiance fields. In *arXiv*, 2023. 3, 8, 11, 12, 15

[18] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *ICCV*, 2023. 3, 4, 5, 8, 12

[19] Peter Hedman and Johannes Kopf. Instant 3D Photography. In *SIGGRAPH*, 2018. 3

[20] Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. Casual 3D Photography. In *SIGGRAPH Asia*, 2017. 3

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2022. 3

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2

[23] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *ICCV*, 2023. 2, 8

[24] Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *ICCV*, 2021. 7, 12

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *SIGGRAPH*, 2023. 3

[26] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson. Pathdreamer: A world model for indoor navigation. In *ICCV*, 2021. 2

[27] Jialu Li and Mohit Bansal. Panogen: Text-conditioned panoramic environment generation for vision-and-language navigation. In *NeurIPS*, 2023. 3

[28] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *ECCV*, 2022. 2

[29] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021. 2

[30] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Learning to generate multiview-consistent images from a single-view image. In *arXiv*, 2023. 3

[31] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 2

[32] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360deg reconstruction of any object from a single image. In *CVPR*, 2023. 3

[33] 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. In *SIGGRAPH*, 2019. 3, 12

[34] 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. 3, 6, 11

[35] 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 *ICCV*, 2023. 3, 12

[36] 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 *CVPR*, 2023. 2, 3, 7, 8, 11, 12

[37] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. Snerf: stylized neural implicit representations for 3d scenes. In *SIGGRAPH*, 2022. 4

[38] Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T Barron, Amit H Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023. 2

[39] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICML*, 2023. 2, 3, 8

[40] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Ben Mildenhall, Nataniel Ruiz, Shiran Zada, Kfir Aberman, Michael Rubenstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. Dreambooth3d: Subject-driven text-to-3d generation. In *ICCV*, 2023. 3

[41] Chris Rockwell, David F Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021. 2

[42] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Ganerf: Leveraging discriminators to optimize neural radiance fields. In *SIGGRAPH Asia*, 2023. 8

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 4, 6, 8, 11, 12

[44] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 3

[45] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, 2021. 7, 12

[46] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. In *arXiv*, 2023. 8

[47] 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, 6, 7, 8

[48] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH Conference Proceedings*, 2023. 6, 7, 12

[49] Luming Tang, Nataniel Ruiz, Qinghao Chu, Yuanzhen Li, Aleksander Holynski, David E Jacobs, Bharath Hariharan, Yael Pritch, Neal Wadhwa, Kfir Aberman, et al. Realfill: Reference-driven generation for authentic image completion. In *arXiv*, 2023. 2

[50] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multiview image generation with correspondence-aware diffusion. In *arXiv*, 2023. 3

[51] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 3

[52] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. In *arXiv*, 2023. 3, 8

[53] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, 2023. 6, 8

[54] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023. 3

[55] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2023. 2

[56] Frederik Warburg*, Ethan Weber*, Matthew Tancik, Aleksander Hołyński, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *ICCV*, 2023. 3, 8, 11, 12

[57] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *arXiv*, 2022. 3

[58] Silvan Weder, Guillermo Garcia-Hernando, Áron Monszpart, Marc Pollefeys, Gabriel Brostow, Michael Firman, and Sara Vicente. Removing objects from neural radiance fields. In *CVPR*, 2023. 3

[59] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 2

[60] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *ICCV*, 2023. 3, 5, 7

[61] 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. 3, 12

[62] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *CVPR*, 2023. 2, 3

Appendix

The appendix includes more details related to the paper.

8. NeRF synthetic prompts

We use the following text prompts for each of the 8 NeRF synthetic datasets from the original NeRF paper [34].

- "chair": "a photo of a chair"
- "drums": "a photo of drums"
- "ficus": "a photo of a ficus plant"
- "hotdog": "a photo of a hotdog"
- "lego": "a photo of a lego bulldozer"
- "materials": "a photo of materials"
- "mic": "a photo of a microphone"
- "ship": "a photo of a ship

9. Datasets

Some of our data can be found online with their respective hyperlinks. Others were created by the authors by scanning with Polycam, and one was obtained by modifying a SPIn-NeRF dataset.

- "dumptruck" (Sketchfab)
- "turtle" (author capture with Polycam)
- "drawing" (Sketchfab)
- "backpack" (SPIn-NeRF dataset)
- "bear" (Sketchfab)
- "billiards" (Sketchfab)
- "norway" (Sketchfab)
- "cat" (author capture with Polycam)
- "boot" (Sketchfab)
- "office" (Sketchfab)

9.1. Marking inpaint regions

To mark inpaint regions, we have a variety of approaches depending on the data. For the "office" scene, we mark inpaint regions as any missing region from the original mesh, found here on Sketchfab. For the other scenes, we place a 3D occluder, such as a cube or cylinders. For the "cat" dataset in Figure 1, we use a cylinder to mark an occlusion of the body and a rectangular prism to mark the tag by the ear. It's quite straightforward to load the mesh into Blender and add the occluders. However, we leave the automatic placement of 3D occlusions for future work. Uncertainty masks from Bayes' Rays [17] or deleted areas from Nerfbusters [56] could provide masks for our method, but their masks are out-of-distribution for SD (Stable Diffusion), shown in Fig. 12 and discussed in Sec. 11.

9.2. Creating NeRF datasets

To create NeRF datasets with the marked inpaint regions, we render the mesh from ∼60 images (64 for all except "backpack" which uses 60). For objects, we render around the object and for e.g., large missing rectangles from a room, we render arcs looking toward the region from different angles, at different elevations. We can mark the inpainting regions by doing a depth check between the actual mesh and the manually placed occluders. To create the "backpack" scene, we took the dataset from SPIn-NeRF [36] and modified the original tight mask around the backpack to include the top part. We also dilated the masks to make them less tight. See Figure 2 for an illustration of our changes to convert it for our scene completion setting rather than object deletion.

10. Additional experiment details

10.1. Inpaint implementation details

Image classifier-free guidance. Some percentage of the time during fine-tuning of the Stable Diffusion model for inpainting [43], everything is masked out. This is similar to dropping out the text prompt with some probability to enable classifier-free guidance. Because of this training strategy, we can enable image-guidance with a formulation similar to InstructPix2Pix [4]. We modify the text-conditioned diffusion inpainting model to predict its score estimate in the form:

$$\epsilon_\phi(z_t, t, c) = \epsilon_\phi(z_t, t, \{c_I, \emptyset\})$$
$$+ s_I \cdot (\epsilon_\phi(z_t, t, \{c_I, c_T\}) - \epsilon_\phi(z_t, t, \{c_I, \emptyset\})) \quad (3)$$
$$+ s_T \cdot (\epsilon_\phi(z_t, t, \{c_I, \emptyset\}) - \epsilon_\phi(z_t, t, \{\emptyset, \emptyset\}))$$

with z_t as the noisy latent at time t, c_I and c_T as conditioning for image & mask, and text respectively. $\emptyset$ means no text or completely masking out the image. s_I and s_T are image and text guidance scales, respectively. We use $s_I > 0$ and $s_T = 0$ to make the model conditioned only on the image and

masked region to inpaint. We note that the popular diffusers[1] library doesn't enable setting $s_I > 0$ *out-of-the-box*, so most works use diffusion inpainting models by using text prompts to describe an inpaint. Trying to describe the scene, however, can be difficult and using image only guidance ("SD Image Cond", where $s_I > 0$ and $s_T = 0$) tends to be more multi-view consistent (Figure 7).

Scheduler details. We use DDIM for our scheduler. We use 20 steps whenever sampling SD [43], regardless of how much noise is added to a current render. This is similar to Instruct NeRF2NeRF [18]'s dataset update procedure. The model we use can be found here as "stabilityai/stable-diffusion-2-inpainting" on Hugging Face.

10.2. NeRF implementation details

For synthetic scenes and objects, we adopt recommended practices for training Nerfacto [48] which is to set the background color to either white or black, disable scene contraction, and turn off the distortion loss. For the "backpack" forward facing scene from SPIn-NeRF [36], we only turn off the distortion loss. Nerfacto is not tuned for LLFF [33] forward-facing scenes, so there may be some artifacts in this dataset compared to our datasets which have larger parallax. For other scenes e.g., those that resemble an indoor room, we train Nerfacto with default settings.

For the baselines besides "Masked NeRF" in Sec. 5.2 "Completing large unknown 3D regions" and Table 2, we train a modified Nerfacto that has losses on patches. Note that "Masked NeRF" is simply Nerfacto, with any modifications as described. The other baselines render 1024 patches of size 32×32 per iteration. An L1 RGB loss and LPIPS [61] loss are applied to these patches, similar to IN2N [18]. Furthermore, we start the dataset update (DU) methods from the result of "Masked NeRF". Each method is trained for 30K iterations, which is typical for Nerfacto.

10.3. Evaluation against inpainted images

We use $t \in [0.4, 1.0]$ noise added to a render before sampling SD and updating a training image. Because $t = 0.4$ is the minimum noise, the inpainter has some freedom to change the inpaint from the current NeRF render. If we added no noise ($t = 0.0$) to the current NeRF render, then the DU (dataset update) methods would be perfect on the NeRF metrics (PSNR, LPIPS, SSIM) because SD would do nothing and the inpainted images would be exactly the NeRF rendered images. Nevertheless, quantifying inpaints without a ground truth result is challenging and our metrics represent an effort to show quantitative evaluation of 3D consistency. We recommend looking at the videos to compare the methods qualitatively.

10.4. Near plane for evaluation metrics

We render with a near plane slightly in front of the training images when reporting our metrics. This is important because NeRFs can cheat by placing "floaters" in front of the training images to explain away any discrepancies in the actual 3D scene compared to the training images. By setting the near plane a bit beyond the camera origin, we can render the true 3D scene without the floaters directly in front of training images.

10.5. Novel-view video metrics

For the *MUSIQ* image quality metric, also used in [35], we take all 300 frames (10 seconds, 30 FPS) of the novel-view videos, downsample each frame by 4x to capture low-frequency structure, and run the MUSIQ [24] model (trained on the AVA dataset) to obtain an image quality score. For the *Corrs* metric, we use LoFTR [45] with a confidence threshold of 0.8. We count the number of correspondences above this confidence threshold for 100 random pairs of frames.

10.6. Metrics for each scene

We provide the individual tables for Table 1 and Table 2 of the paper. See the end of the document for these.

11. Inpainting NeRF casual captures

As mentioned in our limitations section (Sec. 6), Stable Diffusion (SD) fails to produce good inpaints when the mask distribution is similar to those produced by Bayes' Rays [17] or Nerfbusters [56]. In Fig. 12, we illustrate this. Please see the caption for details.

---

[1] https://github.com/huggingface/diffusers

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	5.91	0.70	0.40
LaMask	21.13	0.93	0.23
SD Text Cond	13.27	0.70	0.40
SD Image Cond	13.71	0.75	0.31
Extended Attention	15.20	0.78	0.30
Grid Prior	14.90	0.82	0.27
Joint Multi-View Inpainting	16.59	0.85	0.24

Table 3. **2D inpainting consistency for data "chair".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	5.77	0.66	0.49
LaMask	16.79	0.89	0.39
SD Text Cond	11.17	0.72	0.29
SD Image Cond	12.30	0.76	0.27
Extended Attention	13.23	0.76	0.26
Grid Prior	12.64	0.78	0.27
Joint Multi-View Inpainting	13.25	0.79	0.27

Table 4. **2D inpainting consistency for data "drums".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	6.86	0.73	0.37
LaMask	18.50	0.91	0.22
SD Text Cond	13.16	0.74	0.31
SD Image Cond	13.40	0.76	0.31
Extended Attention	14.86	0.77	0.23
Grid Prior	14.35	0.81	0.28
Joint Multi-View Inpainting	17.24	0.85	0.21

Table 5. **2D inpainting consistency for data "ficus".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	8.84	0.73	0.36
LaMask	21.29	0.92	0.15
SD Text Cond	12.93	0.75	0.35
SD Image Cond	15.12	0.78	0.31
Extended Attention	15.06	0.78	0.30
Grid Prior	15.62	0.84	0.24
Joint Multi-View Inpainting	16.69	0.86	0.24

Table 6. **2D inpainting consistency for data "hotdog".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	8.80	0.73	0.33
LaMask	17.84	0.84	0.18
SD Text Cond	13.01	0.75	0.27
SD Image Cond	14.79	0.79	0.22
Extended Attention	15.42	0.79	0.22
Grid Prior	14.84	0.81	0.21
Joint Multi-View Inpainting	17.89	0.84	0.18

Table 7. **2D inpainting consistency for data "lego".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	8.28	0.72	0.33
LaMask	18.23	0.88	0.16
SD Text Cond	12.80	0.74	0.29
SD Image Cond	14.53	0.78	0.25
Extended Attention	12.85	0.77	0.29
Grid Prior	14.25	0.80	0.22
Joint Multi-View Inpainting	14.53	0.80	0.23

Table 8. **2D inpainting consistency for data "materials".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	6.94	0.73	0.35
LaMask	21.11	0.92	0.13
SD Text Cond	10.97	0.72	0.31
SD Image Cond	13.40	0.77	0.28
Extended Attention	14.26	0.78	0.27
Grid Prior	12.41	0.80	0.26
Joint Multi-View Inpainting	13.26	0.81	0.27

Table 9. **2D inpainting consistency for data "mic".**

	PSNR ↑	SSIM ↑	LPIPS ↓
Masked NeRF	10.66	0.69	0.35
LaMask	21.76	0.82	0.18
SD Text Cond	13.14	0.72	0.31
SD Image Cond	15.95	0.74	0.29
Extended Attention	15.69	0.74	0.29
Grid Prior	16.41	0.78	0.27
Joint Multi-View Inpainting	17.64	0.80	0.24

Table 10. **2D inpainting consistency for data "ship".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	12.45	0.81	0.30	3.82	330
LaMask	27.43	0.95	0.03	3.72	179
SD Image Cond	17.42	0.88	0.15	3.55	305
Inpaint + DU	25.42	0.95	0.06	3.96	260
Ours w/o depth	29.80	0.96	0.03	3.91	218
Ours	29.71	0.96	0.03	3.92	213

Table 11. **Quantitative NeRF baselines for data "cat".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	19.16	0.90	0.15	3.65	164
LaMask	31.66	0.96	0.03	3.66	140
SD Image Cond	21.96	0.89	0.12	3.62	182
Inpaint + DU	28.40	0.94	0.07	3.78	151
Ours w/o depth	29.76	0.94	0.06	3.65	154
Ours	29.74	0.93	0.07	3.69	146

Table 12. **Quantitative NeRF baselines for data "turtle".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	14.20	0.70	0.34	3.95	1083
LaMask	25.45	0.76	0.09	4.03	1040
SD Image Cond	24.23	0.77	0.12	4.00	1024
Inpaint + DU	25.15	0.76	0.13	3.97	1041
Ours w/o depth	26.68	0.82	0.12	4.01	1019
Ours	26.49	0.81	0.13	4.02	1038

Table 13. **Quantitative NeRF baselines for data "drawing".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	11.68	0.79	0.25	3.81	214
LaMask	24.50	0.95	0.05	4.02	195
SD Image Cond	16.45	0.89	0.12	3.66	196
Inpaint + DU	20.42	0.90	0.10	3.80	228
Ours w/o depth	28.76	0.96	0.03	3.84	211
Ours	29.37	0.96	0.03	3.86	176

Table 14. **Quantitative NeRF baselines for data "boot".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	16.86	0.93	0.10	3.82	245
LaMask	27.41	0.96	0.02	3.71	200
SD Image Cond	24.58	0.95	0.04	3.70	261
Inpaint + DU	27.92	0.95	0.03	3.70	245
Ours w/o depth	28.20	0.96	0.03	3.72	259
Ours	28.13	0.96	0.03	3.72	264

Table 15. **Quantitative NeRF baselines for data "bear".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	11.96	0.75	0.29	3.80	193
LaMask	27.73	0.97	0.04	3.84	148
SD Image Cond	19.07	0.87	0.15	3.54	235
Inpaint + DU	28.76	0.95	0.05	3.69	210
Ours w/o depth	27.75	0.95	0.04	3.63	245
Ours	27.48	0.95	0.05	3.62	232

Table 16. **Quantitative NeRF baselines for data "dumptruck".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	16.23	0.70	0.32	3.65	934
LaMask	27.25	0.86	0.09	3.84	865
SD Image Cond	21.58	0.81	0.14	3.98	852
Inpaint + DU	27.75	0.87	0.10	3.83	812
Ours w/o depth	29.54	0.90	0.07	3.65	980
Ours	29.11	0.88	0.07	3.69	1059

Table 17. **Quantitative NeRF baselines for data "norway".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	16.41	0.80	0.19	3.46	1766
LaMask	24.82	0.84	0.04	3.41	1833
SD Image Cond	22.48	0.83	0.07	3.45	1769
Inpaint + DU	24.04	0.84	0.06	3.47	1805
Ours w/o depth	24.38	0.85	0.04	3.39	1847
Ours	23.93	0.83	0.05	3.40	1924

Table 18. **Quantitative NeRF baselines for data "backpack".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	13.37	0.69	0.37	3.50	981
LaMask	25.63	0.84	0.11	3.59	1053
SD Image Cond	24.31	0.83	0.12	3.66	1023
Inpaint + DU	27.13	0.83	0.13	3.69	1078
Ours w/o depth	28.87	0.88	0.09	3.66	1120
Ours	28.78	0.88	0.09	3.66	1137

Table 19. **Quantitative NeRF baselines for data "billiards".**

	PSNR ↑	SSIM ↑	LPIPS ↓	MUSIQ ↑	Corrs ↑
Masked NeRF	14.74	0.75	0.31	3.65	839
LaMask	32.02	0.93	0.03	3.80	779
SD Image Cond	28.26	0.90	0.07	3.64	799
Inpaint + DU	31.05	0.92	0.05	3.70	766
Ours w/o depth	30.35	0.93	0.05	3.68	764
Ours	30.08	0.93	0.05	3.69	768

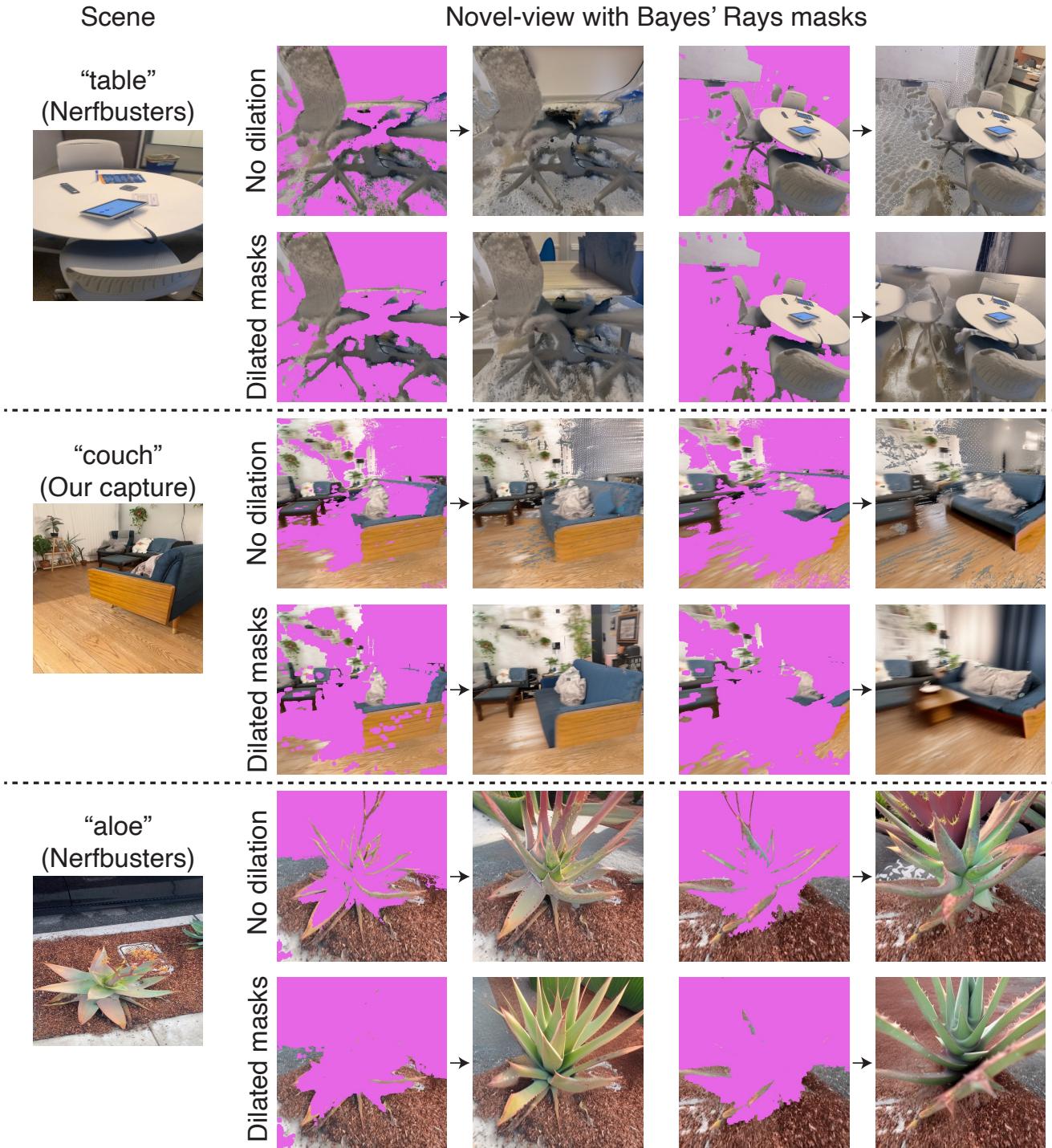Table 20. **Quantitative NeRF baselines for data "office".**

Figure 12. **Out-of-distribution inpaints from Stable Diffusion when applied to casual captures.** On the left, we show a training image from a casually captured scene. On the right, we show masks obtained by rendering a novel-view that has occlusions, and then running Bayes' Rays [17] to delete areas with high uncertainty (marked in pink). We then inpaint these regions with image conditioning. When the mask is not dilated (top rows), the inpaints have many artifacts such as ripple patterns and gray stretches. When the masks are dilated (bottom rows), the inpaints get slightly better but are no longer consistent with the known parts of the scene. This is a challenging setting to address in future work. Retraining SD with masks of this distribution could alleviate the problem, but this is costly and out-of-scope of using an *off-the-shelf* model, as done in our work.