3D Inpainting with Decouplable Gaussian Representation and 2D priors

Anonymous CVPR submission

Paper ID 9390

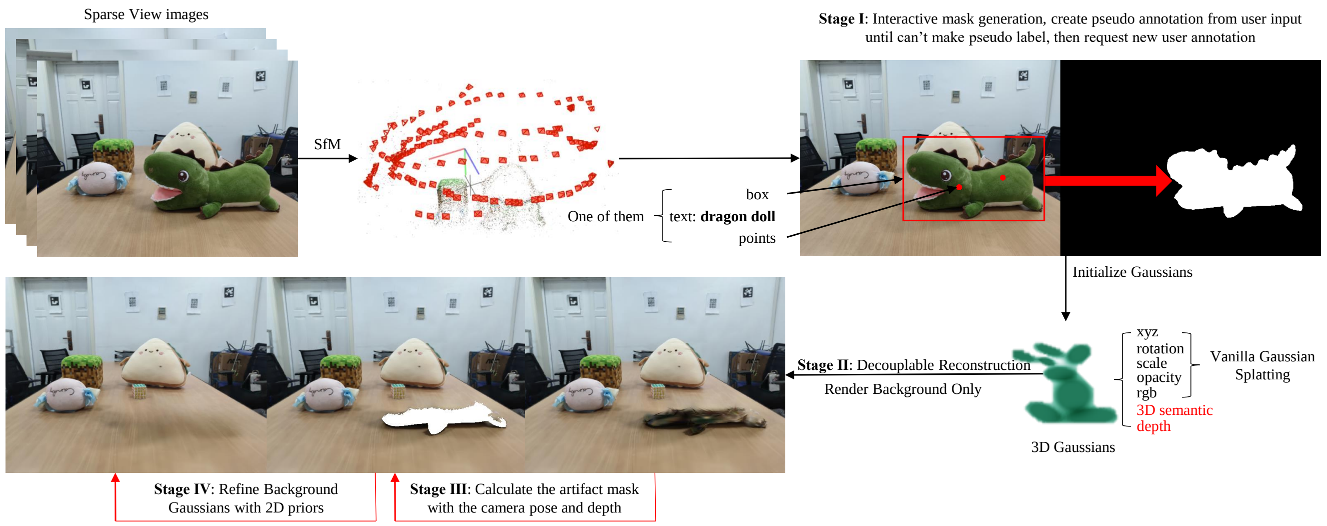
Sparse View images
SfM
Stage I: Interactive mask generation, create pseudo annotation from user input until can't make pseudo label, then request new user annotation
One of them
box
text: dragon doll
points
Initialize Gaussians
xyz
rotation
scale
opacity
rgb
3D semantic
depth
Vanilla Gaussian Splatting
3D Gaussians
Stage II: Decouplable Reconstruction
Render Background Only
Stage IV: Refine Background Gaussians with 2D priors
Stage III: Calculate the artifact mask with the camera pose and depth

Figure 1. An overview of our 3D inpainting framework. We use the SfM method to estimate camera parameters from sparse view images. An interactive mask generation (Sec 4.1) strategy is applied to generate 2D masks from any of the points, boxes, or text prompts. Then, we propose a decouplable Gaussian Splatting (Sec 4.2) to reconstruct the 3D scene and render the background only to get a raw inpainting result. Third, the artifact mask is calculated from 3D geometry (Sec 4.3), and the final inpainting result is obtained by refining the background Gaussians with corrected 2D priors from 2D inpainting (Sec 4.4).

Abstract

Though 2D inpainting has been well-studied in the era of deep learning, 3D inpainting remains challenging. This paper proposes a novel pipeline for inpainting the 3D scene by utilizing the advantages of the 3D reconstruction framework Gaussian Splatting and 2D inpainting models trained on large datasets together. First, we design an interactive process to predict 2D masks with Segment-Anything (SAM). Beginning with user annotation on only one view, our method uses Structure from Motion (SfM) information to generate pseudo annotations for other views and asks users to add annotations if no pseudo ones can be produced for the remaining views. Second, a decouplable Gaussian field is reconstructed from GT images, 2D masks from the first stage, and additional SfM depth supervision. We get raw inpainting results by only rendering the background Gaussians and correcting these results with pre-trained 2D inpainting models. Third, we propose an algorithm leveraging camera pose and stage two depth to find the artifact region in raw inpainting results to ensure concurrent 2D inpainting quality and 3D consistency. Finally, we refine the background Gaussians for minority epochs with the supervision of the corrected inpainting results to get the scene after removal. Additionally, we explicitly derive the gradients with CUDA implementation to maintain the fast speed in Gaussian Splatting. Experiments demonstrate that our method achieves better inpainting quality and can handle challenging scenarios for existing works.

1. Introduction

As the Neural Radiance Field (NeRF) [26] has demonstrated its superiority in the 3D reconstruction task, a series of derivative works [30, 45, 46, 48, 52] have also emerged, including adapting NeRF to 3D editing [9, 28, 43, 44, 47]. 3D inpainting to remove masked portions from a 3D scene,

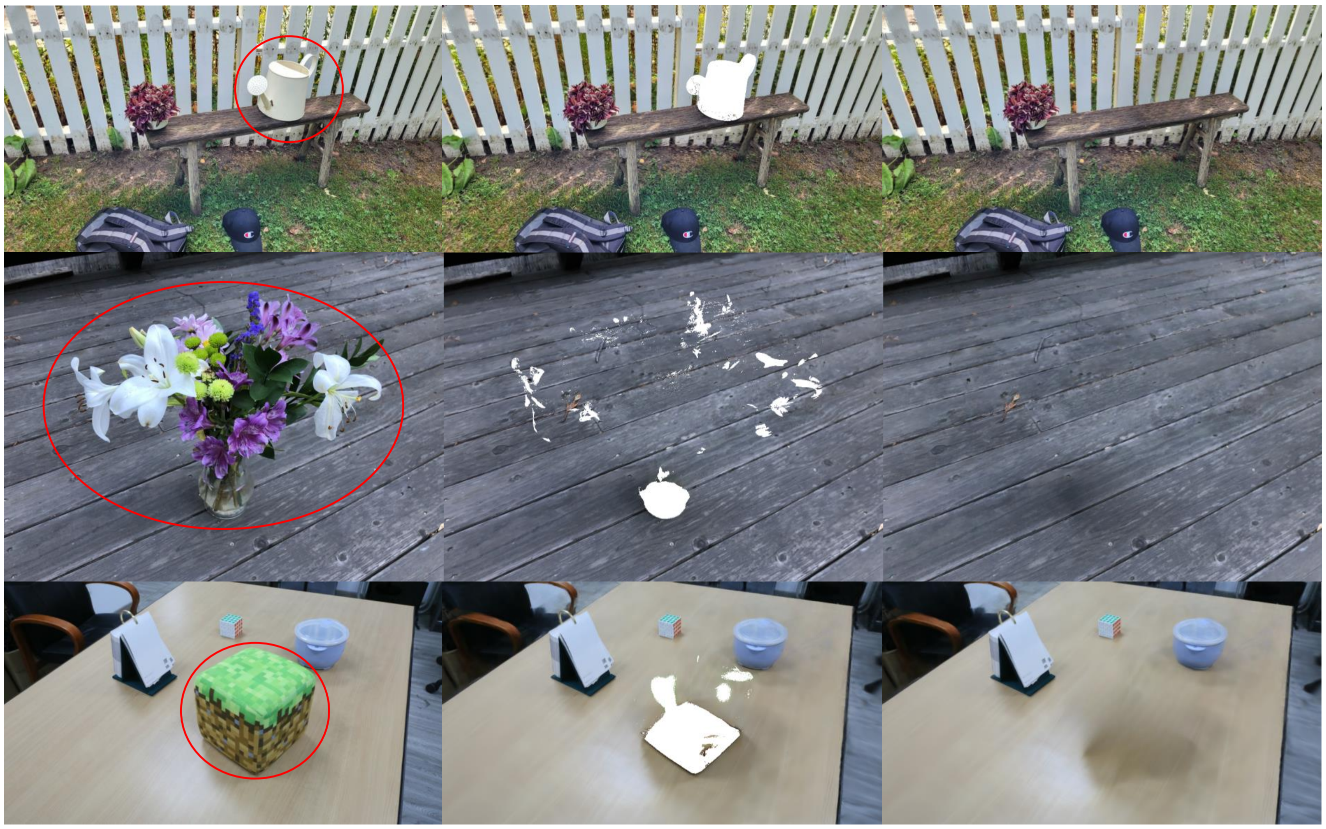
Figure 2. From left to right are GT images, raw inpainting results with artifacts masked, and the final inpainting results with our method.

as an essential component of editing functionality, has also attracted significant research interest. Nevertheless, 3D inpainting remains a challenging task. When using neural radiance fields to reconstruct 3D scenes, one field is specific to that particular scene, and reconstructing another requires training a new neural field from scratch. For reconstruction tasks, the per-scene, per-network paradigm is acceptable as long as sufficient 2D views are available for supervision. However, considering inpainting based on 3D reconstruction means lacking information about the background after deletion, significantly increasing the difficulty of reasonable inpainting.

A natural approach to address this issue is introducing prior knowledge from 2D inpainting. Since 2D inpainting networks [36, 54] are trained on large datasets, they can infer unseen data as well as predict the background texture for 3D inpainting. However, effectively utilizing 2D inpainting is not straightforward. On the one hand, directly applying the mask of the desired region may yield suboptimal results because the portion to be filled is too extensive, even for well-trained 2D inpainting networks. On the other hand, 2D inpainting networks don't consider any 3D information, often leading to 3D inconsistency. In 3D scenes, background pixels covered by the foreground in one view may be visible in another. However, 2D inpainting only provides a potential answer based on its learned data distribution, likely causing conflicts in the 3D context.

While several works have focused on 3D inpainting, they still face the challenge of simultaneously achieving realistic 2D rendering and 3D consistency. The ObjectNeRF [47] and ObjSDF [44] methods achieve editing functionality by decoupling the neural radiance field into the foreground and background. Compared to reconstruction networks, they introduce a learnable object ID and use 2D segmentation maps as supervision to learn this value. Implementing the inpainting functionality involves not rendering the unwanted parts controlled by the object ID. Not surprisingly, due to the absence of the covered background, if these pixels are not visible from any view, their color will be set to the default background color black. Consequently, ”black hole” artifacts appear in the inpainting results. SPIn-NeRF [28], NeRF-Object-Removal [43] and OR-NeRF [50] directly use the results of 2D inpainting [36] as the ground truth to reconstruct the deleted scene. However, as analyzed earlier, these methods do not perform well in complex scenes, especially with occlusions, as they ignore 3D consistency. Additionally, if 2D inpainting produces suboptimal results, improving the render quality is challenging for this kind of method. Figure 3 illustrates the two types of methods' flaws.

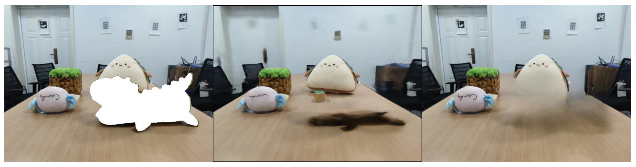
Figure 3. Flaws of the current methods. From left to right are the masked image (left), the inpainting results of ObjSDF [44] (middle) and SPIn-NeRF [28] (right). We can see the ”black hole” artifact in ObjSDF and 3D inconsistency (as the cube is gone) in SPIn-NeRF.

In this paper, we propose a novel 3D inpainting pipeline that leverages the characteristics of 3D reconstruction to maintain 3D consistency and the ability of 2D inpainting to generate background priors simultaneously. Figure 1 is the overview of our method. Our key idea is finding the unseen pixels from all views to bridge 3D reconstruction and 2D inpainting. First (Sec 4.1), we design an interactive mask generation strategy with the 2D segmentation model Segment-Anything (SAM) [14]. To begin with, users annotate an arbitrary image with points, text, or box prompts as long as SAM can provide the desired mask. Then, Structure from Motion (SfM) [34] information will be used to map points in 3D and 2D space. The foreground 3D point cloud will be maintained to aggregate 3d points in each view's mask and generate pseudo annotations for other views. If creating new reliable pseudo annotations is impossible, the user is asked to make a new annotation. The above strategy is made by the observations in practice that annotating once is hard to guarantee all views' masks, while annotating each image is unnecessary as we can utilize 3D information to annotate adjacent views automatically.

Second (Sec 4.2), we turn the SOTA 3D reconstruction method Gaussian Splatting [13] decouplable by adding an attribute to each Gaussian indicating its belongings (foreground or background) and learn this variable through 2D mask supervision. Further, we add depth supervision to the reconstruction process for rendering the depth map used to calculate the artifact mask (Sec 4.3). Raw 3D inpainting results can be gained by rendering background Gaussians only. There will be ”black hole” artifacts caused by the background pixels that are never seen from all the views. Third (Sec 4.3), we utilize the scene depth obtained from the second stage with camera poses to calculate the artifact mask and inpainting this with pre-trained 2D inpainting models [36, 54]. Finally (Sec 4.4), we refine the trained background Gaussians supervised by the corrected raw results. Experiments show that 1/15 epochs of the reconstruction stage is enough for refinement. Additionally, we follow the practice of Gaussian Splatting to explicitly calculate the gradients of our additions to the vanilla version with CUDA implementation.

To conclude, our contributions are:

- A novel 3D inpainting pipeline with better and more robust performance.
- An interactive multiview segmentation strategy that automatically makes pseudo annotations with 3D geometry, balancing precision and human labor with broader scenes.
- A decouplable Gaussian field with CUDA implementation that maintains the speed advantage of vanilla one.
- An artifact mask finding algorithm with depth and camera poses that bridges 3D reconstruction and 2D inpainting.

2. Related Work

2.1. Multiview Segmentation

Though segmentation in 2D [4, 10, 12, 14, 18, 31, 41, 42, 53, 56] is well studied, multiview segmentation for 3D scenes has received less attention despite its non-negligible importance for downstream applications like 3D inpainting. Several self-supervised methods [7, 23] have been proposed, but they have difficulty handling complex scenes, and the mask accuracy is unsatisfactory. Semi-supervised strategies [8, 28, 38, 57] requiring partial labels or user annotations have been proposed to mitigate the challenge. SemanticNeRF [57] observes that partially annotated labels are enough to predict full masks as 3D information is aggregated during training. Specifically, they add a semantic label to the model output and use the exact volume rendering in NeRF [26] to render this label to 2D segmentation maps. Further, SPIn-NeRF [28] constructs a thorough multiview segmentation pipeline with points annotation on a single view. They use one-shot segmentation [10] to estimate an initial mask, followed by a video segmentation [4, 36] to create masks for all views by treating the image sequence as a video. Finally, they follow the practice of SemanticNeRF by changing the reconstruction backbone from NeRF to InstatnNGP [29] to refine the masks. Obviously, SPIn-NeRF consumes considerable resources to achieve multiview segmentation. Besides, OR-NeRF [50] achieves multiview segmentation with points annotation on a single view by mapping points between 2D and 3D with projection. However, this method only works for forward-facing scenes as they do not consider projection errors caused by occlusions.

2.2. 3D Inpainting

Neural Rariance Field has dramatically facilitated the area of 3D scene editing and research [1, 3, 24, 35] focuses on various editing types emerging in large numbers. Works exist for texture editing [5, 45], geometry editing[30, 46, 52], and even enabling multiple functionalities [9, 15–17, 21, 27, 33, 39, 44, 47, 49, 51, 58]. ObjectNeRF [47] and ObjSDF [44] decompose NeRF training into the background and object branches, allowing for rendering specified objects controlled by assigned object IDs. However, they generate ”black holes” at the removal region as there is no supervision for the deletion part during training. SPIn-NeRF [28], NeRF-Object-Removal [43] and OR-NeRF [50] utilize the 2D inpainting [36] method by directly reconstructing the scene after removal with 2D inpainting results as ground truth. Though achieving better rendering quality, these methods suffer from 3D inconsistency and the bottleneck of the 2D inpainting ability. Besides, all the approaches above demand masks for all views, which rely on either unflexible human labor [43] or expensive calculation resources [28]. Additionally, n3f [37] and distilled-feature-fields [15] combine pre-trained language models [4, 18, 32, 37] to enable text editing, thus bypassing the requirement for masks. Still, the rendering quality in the removal region is poor, as no algorithms are designed for "unseen" pixels. Meanwhile, these methods fail if the text prompt cannot accurately locate the desired inpainting region.

3. Background: Gaussian Splatting

Gaussian Splatting [13] is the recent SOTA method for 3D reconstruction. It represents the 3D scene with 3D Gaussians and renders 2D images with rasterization. The 3D Gaussian is defined by its probability density function as:

$$G(x) = e^{-\frac{1}{2}(x)^T\Sigma^{-1}(x)}, \tag{1}$$

where x is the mean of Gaussian, Σ is the covariance matrix for random variable x. In the 3D scene, the Gaussian representation has geometry meaning as x represents the 3D coordinates and Σ is constructed with the scale matrix S and rotation matrix R for the 3D Gaussian ellipsoid: $\Sigma = RSS^TR^T$.

Then, 3D Gaussians are projected [59] to the image plane by $\Sigma^{'} = JW\Sigma W^TJ^T$ for later rendering. Here, W is the viewing transformation, and J is the Jacobian of the affine approximation of the projective transformation. The 2D projection of a 3D Gaussian is then approximated from an ellipse to a circle. After this, the image plane is split into 16x16 tiles. The projected 2D Gaussians in each tile are sorted by their depth. Finally, the image RGBs are calculated by an α blending with the 2D Gaussians overlapping on that pixel, similar to volume rendering:

$$C = \sum_{i\in\mathcal{N}} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j). \tag{2}$$

Here, c_i is the color of each Gaussian, and α_i is given by evaluating a 2D Gaussian with covariance Σ multiplied with a learned per-Gaussian opacity. Finally, the Gaussians are optimized with the classic gradient descent, supervised by RGB values only using the weighted sum of $\mathcal{L}_1$ term and D-SSIM term.

$$\mathcal{L}_{gs} = (1-\lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}. \tag{3}$$

Gradients are explicitly derived to avoid slow computation with automatic differentiation, and the forward and backward passes are implemented with CUDA code.

4. Method

As described before, if we apply 2D inpainting with the full unwanted region mask to get the ground truth of the background, we may encounter 3D inconsistency. On the other hand, if we reconstruct the scene with decouplable neural radiance fields and delete the foreground by rendering the background only, we face "black hole" artifacts. An intuitive solution to this dilemma is reconstruction first, then inpainting the artifacts only. As the artifacts region is the pixels unseen from all views, it can be calculated with 3D geometry by our proposed algorithm. In this section, we first explain how we generate 2D masks interactively in Sec 4.1, second how to turn vanilla Gaussian Splatting decouplable in Sec 4.2, third how to find the mask of the unseen pixels Sec 4.3 and how to refine the Gaussians with 2D inpainting priors in Sec 4.4.

4.1. Stage I: Interactive Mask Generation

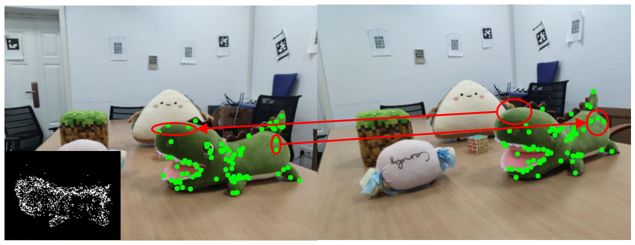

Figure 4. This figure displays a toy demo explaining how our mask generation strategy works. With the views changing from left to right, one more point can be found near the tail. Or three more points can be observed near the head from right to left. As the lower left corner shows, the foreground point cloud P^3 aggregates points from new mask predictions as the view changes and finally becomes accurate.

To begin with, we use SAM [14] to predict masks for the foreground to be inpainted in this stage. Vanilla SAM uses points or boxes as prompts, and we first explain how our method works with points prompts and then describe other prompts like boxes and text.

Given a set of 2D sparse view images $\mathcal{I} = \{I_i\}_{i=1}^{n}$ belonging to a 3D scene, we first reconstruct the camera parameters for this scene with the SfM method. We follow Gaussian Splatting to use COLMAP [34]. COLMAP first extracts features from 2D images and matches each view with the extracted features to form a sparse point cloud. This process typically gives key points with accurate mappings between 2D and 3D space. Formally, assume each image I_i has a set of 2D key points $\{P_i^2\}_{j=1}^{m}$ from COLMAP, where m is the number of points in each image I_i. We can query the corresponding 3D points P_i^3 of 2D key points P_i^2 directly with COLMAP. Intuitively, we can aggregate P_i^3 from all views whose corresponding 2D points P_i^2 are in the mask area to form the foreground's point cloud P^3.

Specifically, we require the user to annotate the desired region for an arbitrary view i with points. The annotation is acceptable as long as SAM can produce the desired mask. Then, we query P_i^3 with P_i^2 in the mask area and initialize P^3 with P_i^3. Then, we query 2D key points on all remaining views with P^3 and choose the view j with the maximum number of 2D key points. We generate a mask for this chosen view j with queried 2D key points as pseudo annotations. We update P^3 with P_j^3 whose corresponding P_j^2 are in the mask of view j. The updation can be done by union P_i^3 and P_j^3. Repeat the above process until we cannot provide pseudo labels for the rest of the views. We request the user to add annotations for one of the remaining views and continue the above process until we handle all the views.

However, using key points only does not guarantee sufficient pseudo-point annotations. This is because the key points numbers are too small compared to the whole image's pixel numbers. Figure 8 in Sec 5.2 also shows a corner case in our method due to the lack of key points in the desired inpainting region. It is possible that many views do not have queriable key points in the mask area. To alleviate this issue, we can use the camera pose to calculate the mappings between 2D and 3D under projection. Given view i's camera pose $[R_i|t_i]$ formed with rotation matrix R_i (Note R_i here is for the camera, and different from the one in Sec 3 for Gaussians) and translation matrix t_i, and camera intrinsic matrix K_i, we can find P_i^3 by projecting 2D points P_i^2 in the mask to 3D:

$$P_i^3 = [R_i|t_i]^{-1}[P_i^2 * d, 1]^T, \tag{4}$$

where d is the depth at pixel P_i^2. For the reverse mapping, we calculate 2D points from corresponding P_i^3 by:

$$P_i^2 = (R_i(P_i^3.(x,y)) + t_i)/P_i^3.z. \tag{5}$$

The problem is that we do not have depth d from SfM alone. As a workaround, we implement the 2D to 3D pass by projecting all 3D key points to the 2D image plane and then choosing the 3D points whose projected 2D pixels are marked as the mask. Another problem is that projection doesn't consider occlusions. 3D points in the background can project into the mask region, so we find 3D points from 2D points by query and calculate the 2D pixel coordinates from 3D points by projection. The 2D to 3D projection is treated as an upper bound for the foreground point cloud (For more details here, please refer to the supplementary). Figure 4 shows how our algorithm aggregates 3D points to form the accurate foreground point cloud.

As for the box prompt, this is straightforward as the first image starts with the box prompt while the rest are the same as points. For the text prompt, users can use 2D open text detection models like GroundingDINO [11, 22] to get the bounding box for the desired region first, then apply the strategies described above to handle the remaining views. As the text prompt can apply to all views directly, we suggest users choose the way that can produce the best segmentation results. Either points or text prompt has drawbacks: points prompt has difficulty processing inadequate views with sharp camera motion, while text fails when it cannot locate the desired region. More discussion can be found in the supplementary.

4.2. Stage II: Decouplable Gaussian Splatting

To decompose the scene into foreground and background, the natural idea is to assign each Gaussian a new attribute indicating its belonging. That is to say, learn a 3D mask for the foreground. Then, we can realize inpainting by simply rasterizing the background Gaussians only. However, previous practices [28, 44, 57] based on NeRF [26] architectures usually create and volume rendering an attribute that has no geometry meaning to the segmentation map. After that, 2D ground truth masks are applied as supervision. Though multiview segmentation has been achieved, the attribute bound to each input point cannot represent the 3D mask.

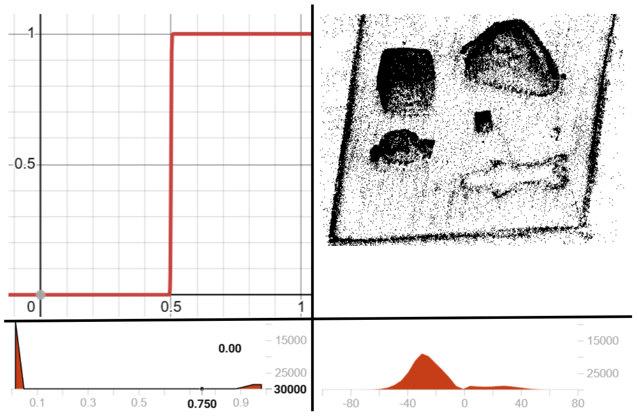

Figure 5. Top left is the function curve of f_{db} with $k = 1000$, it cuts inputs to either 0 or 1 while is differentiable. The top right is the decoupled background Gaussians. We can see that s serves as the 3D mask satisfactorily. The bottom left is the histogram of s with Equation 8 transformation at training step 30000, and the bottom right is without transformation, respectively. We can see that raw s values have the rudiment of binarization, while our proposed algorithm binarizes s successfully.

We create a new attribute s to each Gaussian to meet our demand. We use this s to render 2D segmentation maps by imitating the color blending process in Equation 2 like previous research. At the same time, we try to make this attribute indicate that the Gaussian belongs to the foreground or background. Ideally, s should be either 0 (background) or 1 (foreground), while it can be rasterized to the 2D segmentation map for loss computation. First, we add s to Gaussians and render the 2D segmentation map by changing the color c_i in Equation 2 to s_i:

$$S = \sum_{i \in \mathcal{N}} s_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{6}$$

where S is the segmentation map. We optimize s with binary cross entropy loss $\mathcal{L}_{bce}$:

$$\mathcal{L}_s = \mathcal{L}_{bce}(S, S_{gt}), \tag{7}$$

here S_{gt} is the ground truth 2D segmentation map obtained from Sec 4.1. We investigated the distribution of the trained s under this setting and found that it is already roughly binary with biased dwarf peaks (see Figure 5). So, an easy way is to constrain s to $(0, 1)$ first and then apply a binarization with a threshold of 0.5. However, binarization is not differentiable. With the experience from previous works [19, 20, 44], we apply a differentiable binary function f_{db} to s before it goes to Equation 6:

$$s = f_{db}(f_{sigmoid}(s)), \text{where } f_{db}(x) = \frac{1}{1 + e^{-k(x-t)}}, \tag{8}$$

here, k is a hyper-parameter to control the "sharpness" of the f_{db} function while t is the threshold for binarization. We first use the Sigmoid function to constrain s in $(0, 1)$ and set the threshold of the differentiable binary function f_{db} to 0.5. We can now optimize s with Equation 7 to render a 2D segmentation map and learn a 3D mask simultaneously. Figure 5 shows the features of f_{db} and the performance of our proposed s.

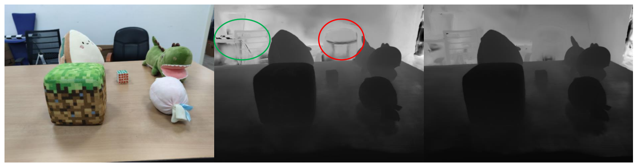
Figure 6. From left to right are GT, depth map without and with supervision, respectively. The part in the green circle shows higher quality, while we can see over smoothness in the red circle.

Go one step further to find the "black hole" artifact mask. Recall that a 2D to 3D mapping needs depth information, and occlusions in 3D geometry are also closely related to depth. We try to render the depth map in the reconstruction stage for future usage in Sec 4.3. We obtain the depth d of each Gaussian with a slightly change to Equation 5:

$$d = R_i(P_i^3).z + t_i. \tag{9}$$

Note that Gaussians are not like points. They have shapes and occupy a specific space. We can't use the depth of Gaussian as depth for the pixel directly. Therefore, we render the 2D depth map like s Equation 6:

$$D = \sum_{i \in \mathcal{N}} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{10}$$

where D is the depth map. Experiments show that only forward rendering without supervision can produce reasonable results already (see Figure 6). Additionally, we follow DSNeRF [6] to make a COLMAP depth supervision:

$$\mathcal{L}_d = w * (D - D_{gt})^2, \tag{11}$$

here, w is the weights calculated pixel-wise from COLMAP's projection error, and D_{gt} is the ground truth depth from COLMAP. As COLMAP can only provide discrete depth (actually, only key points have depth), we multiply a weights mask $\mathcal{M}$ to $\mathcal{L}_d$ in which pixels without GT depth will set to 0, and others to w. Figure 6 shows the performance of the added depth supervision.

To summarize, we rewrite the color loss used in Gaussian Splatting Equation 3 for clarity as:

$$\mathcal{L}_c = (1 - \lambda)\mathcal{L}_1(C, C_{gt}) + \lambda \mathcal{L}_{D-SSIM}(C, C_{gt}), \tag{12}$$

where C is the rendered 2D image calculated by Equation 2 and C_{gt} is the ground truth color. And our total loss L for decouplable reconstruction is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_c + \lambda_2 \mathcal{L}_s + \lambda_3 \mathcal{L}_d. \tag{13}$$

λ_1, λ_2, and λ_3 are hyper-parameters for each loss's weight. To maintain vanilla Gaussian Splatting's fantastic speed, we explicitly derive gradients for s and d and integrate them into the CUDA implementation. The supplementary provides more details.

4.3. Stage III: Artifact Mask Calculation

With a decoupable Gaussian field, we can get raw inpainting results by only rasterizing the background Gaussians. Unsurprisingly, the raw results show "black holes" like ObjSDF and ObjectNeRF, as we didn't consider background texture before. Note that the "black hole" artifacts mean these pixels are unseen from all available views. Thus, the reconstruction method gives it the default background value as black. In other words, if the masked pixel in target view i can be seen from another source view j, the 3D model can learn this pixel value in target view i regarding source view j, and the "black hole" artifact will not happen to this pixel. Formally, we project pixels in the non-mask region from all views to 3D space to get 3D points collection $\mathcal{P}^3 = \{P_i^3\}_{i=1}^n$ using Equation 4, and depth can be gained from Equation 10 as described in Sec 4.2. To continue, we traverse the image set $\mathcal{I} = \{I_i\}_{i=1}^n$ one by

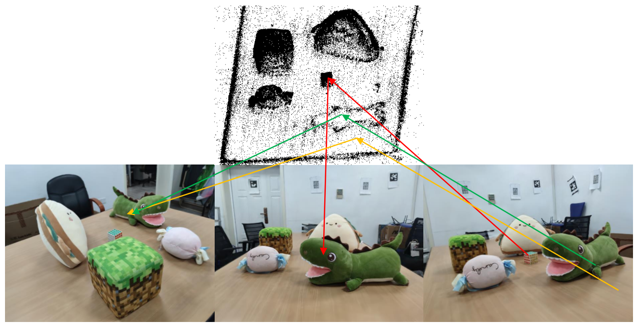
Figure 7. Suppose the bottom right view is the source while the left and middle are targets. The dragon is the desired inpainting region with masks omitted. As the red line shows, we can find the non-mask pixel on the cube from the source view falls in the masked region of the middle view. The green line can serve as a reference. However, although the pixel indicated by the yellow line can also be found in the masked region of the left view, it cannot be a reference as its depth in the left view misaligns with the background depth. The green line indicates the actual pixel we need, which is, unfortunately, covered by all views.

one and mark each image as the target view. For each target view I_i, we traverse $\mathcal{P}^3 = \{P_j^3\}_{j=1}^n$ to project P_j^3 to I_i with Equation 5 and Equation 9 to get the 2D projected pixel coordinates and depth, respectively. Suppose the projected pixel is in the mask area of the target view I_i, and the corresponding projected depth is under a certain threshold compared to the background depth at this pixel. In that case, we can confirm that the pixel from the source view j can be the reference of its corresponding pixel in the target view i. The background depth can be obtained with Equation 10 by considering only the background Gaussians decided by s described in Sec 4.2. After the loop has been completed, the pixels left behind with no reference pixels will form the "black hole" region. In Figure 7, we draw a toy example to show how to find the artifact mask.

4.4. Stage IV: Refine with 2D Priors

Once we find the mask of the artifact region, we can feed this mask and the raw inpainting results to a 2D inpainting model [36, 54] to get background priors. Suppose the corrected 2D results are C'_{gt}. Besides, we can inpainting depth maps similarly to get D'_{gt}. Additionally, we follow SPIn-NeRF's [28] experience to add a perceptual loss [55] term $\mathcal{L}_p$ to the inpainting area to expect not totally the same reconstruction, but perceptual similar RGBs. Combine above, we refine the decoupled Gaussian field from background Gaussians:

$$\mathcal{L} = \lambda_1 \mathcal{L}_c(C, C'_{gt}) + \lambda_2 \mathcal{L}_2(D, D'_{gt}) + \lambda_3 \mathcal{L}_p(C, C'_{gt}). \quad (14)$$

λ_1, λ_2 and λ_3 are hyper-parameters for each loss's weight. Naturally, we don't need the semantic loss term during the refinement as all Gaussians are background. We use L_2 loss to optimize the depth map as we already obtain a complete depth image. More details are in the supplementary.

5. Experiments

5.1. Settings

We select 37 scenes from various commonly used 3D datasets, including NeRF [26], IBRNet [40], LLFF [25], MipNeRF360 [2] and SPIn-NeRF [28]. As for metrics, we adopt pixel-wise accuracy (Acc) and intersection over union (IoU) to assess our masks. Due to the lack of ground truth for the 3D inpainting task, we perform quality analysis for our inpainting results.

5.2. Mask Generation

Table 1 shows that our mask generation method is better than SPIn-NeRF considering accuracy and IoU. We omit the "book" scene from the average calculation as we find the ground truth of this scene is not accurate. The low metrics for "trash" are because there are nearly no key points at the trash bin's lower half region (see Figure 8). This also reveals the bottleneck of our mask generation method mentioned in Sec 4.1. However, we may get accurate masks if we apply text prompts to challenging scenes with point prompts for all views. Another concern is the human labor needed at this stage. Most scenes with dense camera trajectories can automatically annotate all the views except for the initial annotation. Only scenes from the NeRF Synthetic dataset have views where the desired foreground region can be unseen, which needs human judgment. However, human annotations will not exceed 10% for these scenes.

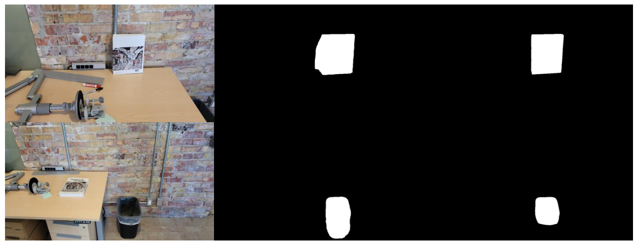
Figure 8. From the left to the right are the images, GT masks, and our results. Upper is the "book" scene, and lower is the "trash" scene.

5.3. Decouplable Reconstruction

First, as mentioned in Sec 5.2, masks generated by Sec 4.1 can be inaccurate sometimes if key points are inadequate in the desired region or SAM [14] cannot produce an accurate mask with unbalanced pseudo annotation. However, Figure 9 shows that our decouplable reconstruction can repair the partial mask after training.

Second, our depth supervision can regularize the scene with a large region covered by the same texture, like walls or mirrors, as an additional benefit besides calculating the

Table 1. Comparison of mask between our interactive method (Sec 4.1) and SPIn-NeRF [28]. The first row indicates the scene name.

	1	2	3	4	7	9	10	12	trash	Mean	SPIn-NeRF[28]	book
acc↑	99.80	99.82	99.73	99.79	99.81	99.78	99.87	99.30	99.51	99.71↑	98.91	98.99
IoU↑	96.77	96.47	97.48	98.50	97.43	96.29	95.47	91.73	88.68	95.42↑	91.66	83.07

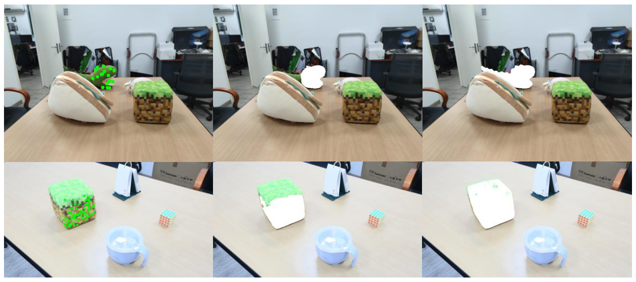
Figure 9. From left to right are the image with pseudo annotations, mask predicted by SAM, and mask rendered by our decouplable Gaussian field. We can see that even if SAM generates a partial mask with unbalanced point annotations, it can be repaired in the later reconstruction stage. Though the result of the second row is imperfect, it can be post-processed with dilation.

artifact mask. Figure 10 illustrates this phenomenon. This is perhaps because depth supervision adds geometry information to Gaussian Splatting to avoid the ambiguity of a large, continuous, smooth region.

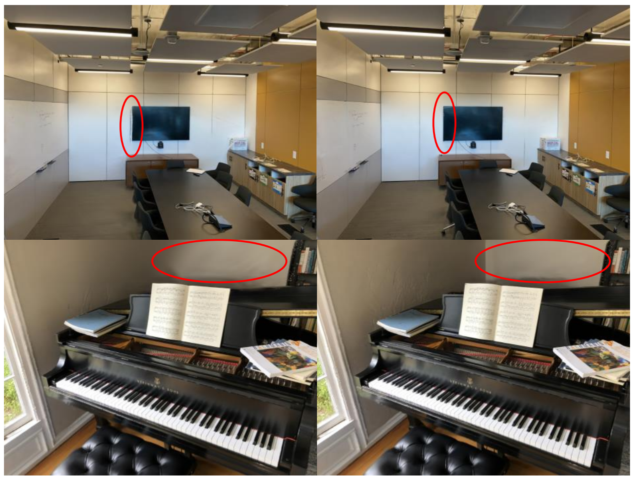
Figure 10. Left is the results from original Gaussian Splatting directly, and right is the results of our method. We can see the left edge of the TV on the wall warped, and the circled wall area is blurred in the original implementation. With our depth supervision, both situations have improved.

Finally, experiments show that our addition to the vanilla Gaussian Splatting can be trained without an apparent speed drop. Table 2 shows an approximate time analysis between our method and vanilla Gaussian Splatting.

Table 2. Approximate time comparison between ours and the vanilla version. As time varies for different scenes, we provide data from the same scene with RTX 4090 GPU.

	Gaussian Splatting	ours
training time (30000 steps)	7m0s	7m3s
refine time (2000 steps)	/	1m5s

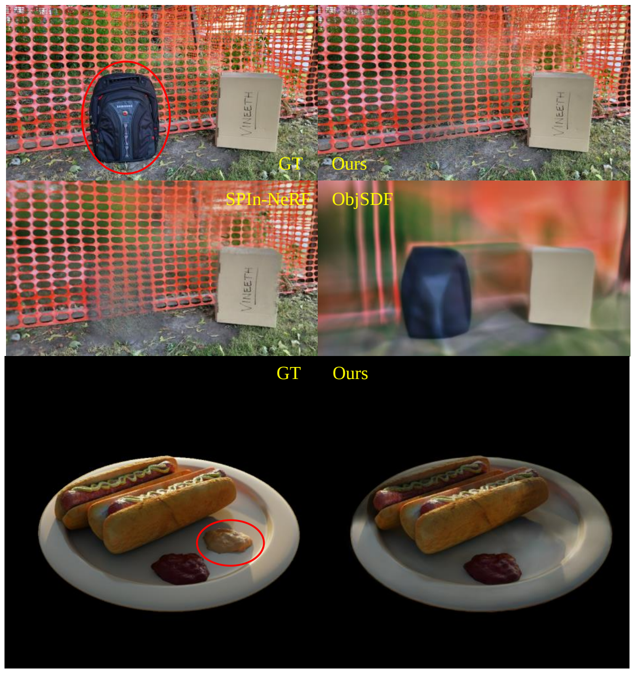

Figure 11. In the above scene, though SPIn-NeRF can give a result, our rendering quality is higher. ObjSDF fails as it cannot converge on this scene. For the below scene, even generating masks is a big problem for the current methods. Though we provide masks for SPIn-NeRF and ObjSDF, they fail due to unstable training.

5.4. Refined inpainting

Figure 11 shows examples where the previous methods fail or perform poorly, but our method performs robustly or better. Also, we display part of our inpainting results as Figure 2 shows. Please refer to the supplementary for the complete experiment results.

6. Conclusion

This paper proposes a novel 3D inpainting pipeline with better performance than previous methods. However, artifacts still exist, especially for forward scenes where we cannot acquire 3D reference pixels and errors accumulated in our pipeline, as explained in the supplementary.

References

[1] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. SINE: Semantic-driven Image-based NeRF Editing with Prior-guided Editing Field, 2023. 3

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5460–5469, 2022. 7

[3] Sagie Benaim, Frederik Warburg, Peter Ebert Christensen, and Serge Belongie. Volumetric Disentanglement for 3D Scene Manipulation, 2022. 3

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. 3, 4

[5] Zhiqin Chen, Kangxue Yin, and Sanja Fidler. AUV-Net: Learning Aligned UV Maps for Texture Transfer and Synthesis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1455–1464, 2022. 3

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12872–12881, 2022. 6

[7] Zhiwen Fan, Peihao Wang, Yifan Jiang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. NeRF-SOS: Any-View Self-supervised Object Segmentation on Complex Scenes, 2022. 3

[8] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic NeRF: 3D-to-2D Label Transfer for Panoptic Urban Scene Segmentation, 2022. 3

[9] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P. J. Narayanan. Interactive Segmentation of Radiance Fields, 2023. 1, 3

[10] 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 *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1551–1560, 2021. 3

[11] IDEA-Research. Grounded-sam. `https://github.com/IDEA-Research/Grounded-Segment-Anything`, 2023. 5

[12] Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan Liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Segment Anything in High Quality, 2023. 3

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 3, 4

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything, 2023. 3, 4, 7

[15] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for Editing via Feature Field Distillation. In *Advances in Neural Information Processing Systems*, pages 23311–23330, 2022. 3, 4

[16] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. PaletteNeRF: Palette-based Appearance Editing of Neural Radiance Fields, 2023.

[17] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation, 2022. 3

[18] Boyi Li, Kilian Q. Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven Semantic Segmentation. In *ICLR*, 2022. 3, 4

[19] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-Time Scene Text Detection with Differentiable Binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11474–11481, 2020. 6

[20] Minghui Liao, Zhisheng Zou, Zhaoyi Wan, Cong Yao, and Xiang Bai. Real-Time Scene Text Detection With Differentiable Binarization and Adaptive Scale Fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1): 919–931, 2023. 6

[21] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing Conditional Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5753–5763, 2021. 3

[22] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 5

[23] Xinhang Liu, Jiaben Chen, Huai Yu, Yu-Wing Tai, and Chi-Keung Tang. Unsupervised Multi-View Object Segmentation Using Radiance Field Propagation. In *Advances in Neural Information Processing Systems*, pages 17730–17743, 2022. 3

[24] Aryan Mikaeili, Or Perel, Daniel Cohen-Or, and Ali Mahdavi-Amiri. SKED: Sketch-guided Text-based 3D Editing, 2023. 3

[25] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 7

[26] 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 *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pages 405–421, 2020. 1, 3, 5, 7

[27] Ashkan Mirzaei, Yash Kant, Jonathan Kelly, and Igor Gilitschenski. LaTeRF: Label and Text Driven Object Radiance Fields. In *Computer Vision – ECCV 2022: 17th Eu-*

ropean Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III, pages 20–36, 2022. 3

[28] 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, 2023. 1, 2, 3, 4, 5, 7, 8

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41 (4):102:1–102:15, 2022. 3

[30] Yicong Peng, Yichao Yan, Shengqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. CageNeRF: Cage-based Neural Radiance Field for Generalized 3D Deformation and Animation. In *Advances in Neural Information Processing Systems*, pages 31402–31415, 2022. 1, 3

[31] Lu Qi, Jason Kuen, Tiancheng Shen, Jiuxiang Gu, Wenbo Li, Weidong Guo, Jiaya Jia, Zhe Lin, and Ming-Hsuan Yang. High Quality Entity Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4047–4056, 2023. 3

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763, 2021. 4

[33] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned Radiance Fields from a Single View. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8948–8958, 2021. 3

[34] Johannes Lutz Schönberger, True Price, Torsten Sattler, Jan-Michael Frahm, and Marc Pollefeys. A vote-and-verify strategy for fast spatial verification in image retrieval. In *Asian Conference on Computer Vision (ACCV)*, 2016. 3, 4

[35] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-E: Text-guided Voxel Editing of 3D Objects, 2023. 3

[36] 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 *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3172–3182, 2022. 2, 3, 4, 7

[37] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D Distillation of Self-Supervised 2D Image Representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453, 2022. 4

[38] Matthew Wallingford, Aditya Kusupati, Alex Fang, Vivek Ramanujan, Aniruddha Kembhavi, Roozbeh Mottaghi, and Ali Farhadi. Neural Radiance Field Codebooks. In *ICLR*, 2023. 3

[39] Bing Wang, Lu Chen, and Bo Yang. DM-NeRF: 3D Scene Geometry Decomposition and Manipulation from 2D Images. In *ICLR*, 2023. 3

[40] 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. 7

[41] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking Everything Everywhere All at Once, 2023. 3

[42] Xinlong Wang, Xiaosong Zhang, Yue Cao, Wen Wang, Chunhua Shen, and Tiejun Huang. SegGPT: Segmenting Everything In Context, 2023. 3

[43] Silvan Weder, Guillermo Garcia-Hernando, Aron Monszpart, Marc Pollefeys, Gabriel Brostow, Michael Firman, and Sara Vicente. Removing Objects From Neural Radiance Fields, 2022. 1, 2, 4

[44] Qianyi Wu, Xian Liu, Yuedong Chen, Kejie Li, Chuanxia Zheng, Jianfei Cai, and Jianmin Zheng. Object-Compositional Neural Implicit Surfaces. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 197–213, 2022. 1, 2, 3, 5, 6

[45] Fanbo Xiang, Zexiang Xu, Miloš Hašan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. NeuTex: Neural Texture Mapping for Volumetric Neural Rendering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7115–7124, 2021. 1, 3

[46] Tianhan Xu and Tatsuya Harada. Deforming Radiance Fields with Cages. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 159–175, 2022. 1, 3

[47] 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 *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13759–13768, 2021. 1, 2, 3

[48] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. NeuMesh: Learning Disentangled Neural Mesh-Based Implicit Field for Geometry and Texture Editing. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pages 597–614, 2022. 1

[49] Weicai Ye, Shuo Chen, Chong Bao, Hujun Bao, Marc Pollefeys, Zhaopeng Cui, and Guofeng Zhang. IntrinsicNeRF: Learning Intrinsic Neural Radiance Fields for Editable Novel View Synthesis, 2023. 3

[50] Youtan Yin, Zhoujie Fu, Fan Yang, and Guosheng Lin. Ornerf: Object removing from 3d scenes guided by multiview segmentation with neural radiance fields, 2023. 2, 3, 4

[51] Hong-Xing Yu, Leonidas Guibas, and Jiajun Wu. Unsupervised Discovery of Object Radiance Fields. In *ICLR*, 2022. 3

[52] 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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18332–18343, 2022. 1, 3

[53] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster Segment Anything: Towards Lightweight SAM for Mobile Applications, 2023. 3

[54] Lvmin Zhang and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models, 2023. 2, 3, 7

[55] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 7

[56] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast Segment Anything, 2023. 3

[57] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15818–15827, 2021. 3, 5

[58] Jingsen Zhu, Yuchi Huo, Qi Ye, Fujun Luan, Jifan Li, Dianbing Xi, Lisha Wang, Rui Tang, Wei Hua, Hujun Bao, and Rui Wang. I$\hat{2}$-SDF: Intrinsic Indoor Scene Reconstruction and Editing via Raytracing in Neural SDFs, 2023. 3

[59] M Zwicker, H Pfister, J van Baar, and M Gross. EWA volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, pages 29–538, 2001. 4