NeRaser: NeRF-based 3D Object Eraser

Liuxin Qing* Shao Zhou* Shi Chen* Xichong Ling*
{liuqing, zhousha, shichen, lingxi}@student.ethz.ch

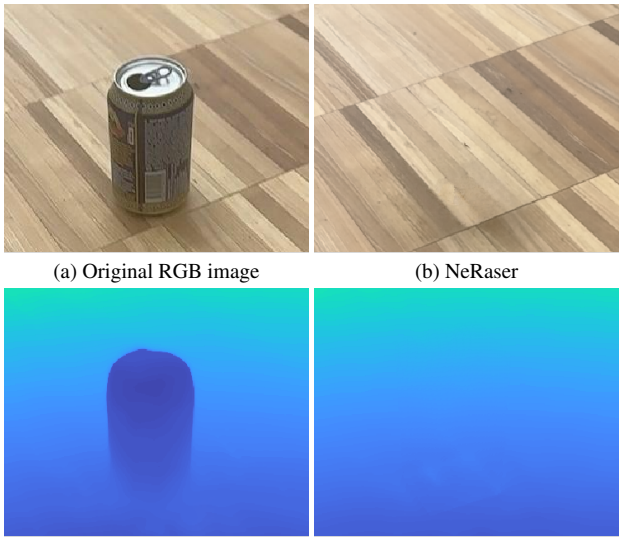
Abstract

We introduce a novel approach for object removal in a 3D Neural Radiance Field representation (NeRF). Different from most existing video inpainting methods which inpaint individual 2D frames, our proposed methodology exploits a rich 3D understanding of the scene to ensure visual consistency of the area occluded by the removed object during novel view synthesis, offering a more robust and visually coherent solution for object removal from videos. We showcase the robustness and coherency of our approach through comprehensive experiments and demonstrate its potential applications in enhancing immersive experiences and content creation in mixed-reality environments.

1. Introduction

Video inpainting aims to fill occluded regions of a video with visually consistent content while ensuring spatial and temporal consistency. This technique holds wide applications across diverse fields, including visual artistry, architectural visualization, and mixed reality. Recently, deep learning-based approaches have shown remarkable progress in this task. However, most mainstream methods are based on frame-wise image inpainting of the entire occluded area in individual frames and do not utilize the fact that part of the occluded area in one frame may be visible in other frames. Not only would this result in inefficiency by having to inpaint a larger area than necessary in every individual frame, but the results would also frequently appear visually inconsistent when viewed as a sequence, as the inpainting procedure does not impose any spatial consistency constraints. Neural Radiance Fields (NeRF) methods, capable of producing high-fidelity scene representations and generating novel views, have been gaining popularity in reconstruction tasks. The generalization capabilities and geometric information encoded in NeRF can help with object removal style video inpainting, by filling in the occluded area using a learned scene representation, thus minimizing the area to be inpainted and promising enhanced visual consistency.

*These authors contributed equally to this work



(a) Original RGB image (b) NeRaser

(c) Original Depth image (d) NeRaser

Figure 1. **A demonstration of NeRaser.** The data is from a scene self-captured using Polycam. A teaser video can be accessed via this URL

The main contributions of our work are the introduction of a novel NeRF-based inpainting method based on NeRF representation that maximizes the use of geometric information while minimizing inpainting, and an accompanying, practical pipeline accessible to ordinary users. Our method takes advantage of multi-view perception to minimize the inpainted regions, improving the visual consistency of the inpainted pixels under different viewing angles.

2. Related Work

2.1. Semantic Segmentation

Semantic segmentation partitions an image into semantically meaningful regions. Traditional methods utilized handcrafted features and algorithms like thresholding, edge detection, and region-based segmentation such as Watershed [15]. With the advent of deep learning, Convolutional Neural Networks (CNNs) have revolutionized semantic segmentation. As one such example, Cheng et al. [5] present a modular interactive VOS (MiVOS) framework that decou-

ples interaction-to-mask and mask propagation, allowing for higher generalizability and better performance in propagating masks along temporal sequences.

2.2. Inpainting

Given an image and a mask of marking specific regions in the image, inpainting aims to fill the masked region with entirely new pixels, generated in a way that is semantically coherent with the rest of the image. A variety of deep learning-based generative methods have been proposed for image inpainting. Pathak et al. [12] build an encoder-decoder generative adversarial network based on U-Net, while Wang et al. [19] propose an external-internal inpainting scheme with a monochromatic bottleneck that helps remove the inpainting artifacts. Inpainting can also be extended to video, but naively inpainting each individual frame discards valuable information encoded in the scene geometry and often introduces visual artifacts that appear inconsistent when viewed sequentially. Liu et al. [7] introduce a model that maintained consistency over object movement across temporal sequences along with texture similarity within a single frame. A recent work [21] based on the propagation method integrates image and feature warping, and thus achieves good performance with more efficient memory utilization. To our knowledge, no video inpainting methods extend to novel view synthesis, limiting their application in mixed-reality settings.

2.3. Inpainting in NeRF

Neural Radiance Fields (NeRF) introduced by Mildenhall et al. [8] presented a novel approach for scene representation and rendering. Many extensions have since been proposed. Instant-NGP [11] uses occupancy grids and an effective multi-resolution hash encoding for an accelerated model. Mip-NeRF [3] reduces objectionable aliasing artifacts and improves the ability to represent fine detail by rendering anti-aliased conical frustums instead of rays. Mip-NeRF 360 [4] utilizes non-linear scene parameterization, online distillation, and a novel distortion-based regularizer to successfully synthesize unbounded scenes.

However, perhaps due to the implicit nature of NeRF representations, few works focus on editing NeRF scenes by removing objects. Weder et al. [20] learn a NeRF representation of scenes with removed objects by using a confidence-based view selection procedure to select geometrically consistent 2D inpainted images for training, but still make insufficient use of scene geometry when inpainting. SPin-NeRF [9] generates a 3D segmentation mask for a target object and distills its information into 3D space. It ensures multi-view consistency by introducing a perceptual loss over the inconsistently inpainted images, rather than ensuring that the inpainted 2D RGB priors are consistent to begin with. InpaintNeRF360 [18] applies depth-space warping to multiview test-guided segmentation masks to enforce viewing consistency. NeRF-In [6] transfers user masks to multiple views and estimates color and depth in these areas, then jointly inpaints images across views by updating NeRF model parameters. Although these existing works have attained a degree of consistency, there is an over-reliance on the 2D inpainting quality and quantity. Our method confronts this issue directly, minimizing the 2D inpainting area and number of inpainted frames. Moreover, our approach places no particular constraint on the underlying NeRF algorithm as long as depth loss is included, and therefore can easily adapt to future improvements to NeRF.

2.4. Nerfstudio

Our project is based on an open-source modular framework for NeRF development called NeRFStudio [17], which combines recent progresses in NeRF to achieve a balance between speed and quality, while also remaining flexible to future modifications. It includes utilities for many common tasks in NeRF development, such as dataset preparation, training, logging, and rendering, to expedite the development of core algorithms.

3. Method

Our 3D representation is mainly based on an integrated module in nerfstudio called depth-erfacto [17], which incorporates camera pose refinement, hash encoding, and proposal sampling to facilitate learning from mobile phone recorded datasets.

Figure 2 illustrates the procedures of our algorithm. Note that we assume that the target object is placed on a locally flat surface, and ideally the surface should only include low-frequency textures. A 360° scan of the object is also preferred.

3.1. Data Preparation

Our dataset is composed of ordinary objects in indoor scenes, which can be easily recorded by a smartphone with a depth sensor. Three attributes for each frame are required for our inpainting process: a dense depth map of the scene, extrinsic and intrinsic camera parameters, and the masks of our target object. The former two can be acquired using a smartphone and a 3D capture application such as Polycam [1]. We use Polycam to scan the scene, produce image sequences, and obtain the corresponding camera poses and dense depth maps. The masks of a target object are generated with the MiVOS [5] framework, which offers an interactive interface where users can select the object in the first frame and refine the mask contours with a few clicks. When a satisfactory object mask in the first frame is ready, MiVOS automatically propagates the mask throughout the entire image sequence.

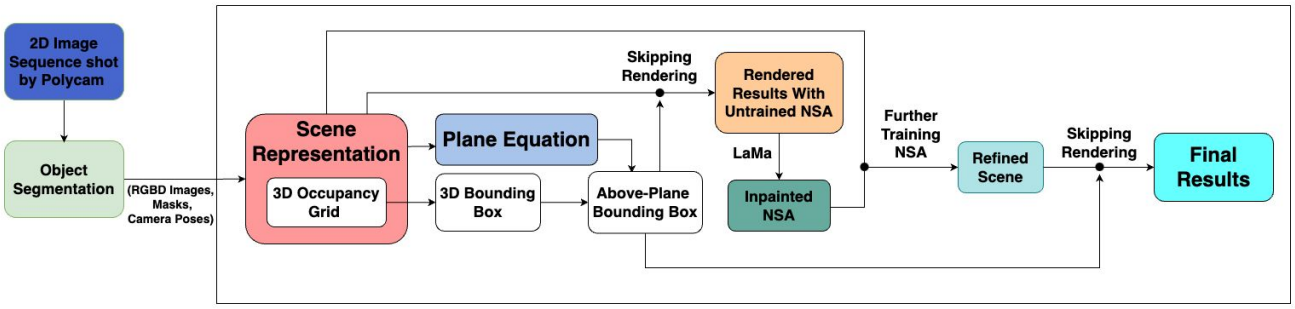


Figure 2. **Overview of the NeRaser pipeline.** A Sequence of 2D images is fed into space-time memory networks to generate 2D masks for the object, which together with the RGBD images and camera poses form the input to our method. We then compute a 3D occupancy grid based on the 2D masks and simultaneously estimate the 3D plane of the ground upon which the object was placed. This allows us to compute a tight bounding box which we use to skip ray samples during two sequential training steps. This leads to the object being removed during the first training stage, leaving a never-seen-area (NSA) on the surface. We then inpaint the NSA using LaMA [16]. Following this, the NeRF is further trained using rendered images with the NSA inpainted, leading to a more refined scene where the object is completely eliminated.

3.2. First training stage

We use nerfstudio [17], an integrated framework for training NeRF models as our codebase. A 3D scene representation of the environment surrounding the target object can be obtained with image sequences and corresponding camera poses. The implicit scene representation is not only the pillar of our reconstruction results but also of crucial significance to the following task of plane estimation.

3.3. Estimation of the Occupancy Grid

We introduce the idea of occupancy grids in order to acquire a fine-grained spatial representation of the target object we want to remove.

Let $V = \{v_1, v_2, \ldots, v_n\}$, $v_i \in \mathbb{R}^3$ be the set of vertices of the 3D grid. For each image I_i in a total of N images, we project the vertices V onto the corresponding 2D mask image. The projection operation, denoted as P_i, maps each 3D vertex v to a 2D point on the mask of the i-th image.

We calculate the occupancy value of vertex V in the i-th image as

$$O_i(v) = \begin{cases} 1 & \text{if } P_i(v) \in \text{Mask of } I_i \\ 0 & \text{otherwise} \end{cases}$$

where $P_i(v)$ denotes the projected 2D point using the projection operation P_i.

To determine the final set of occupied vertices, we compute the average occupancy and use a simple threshold:

$$G = \left\{ v \in V \;\middle|\; \frac{1}{N} \sum_{i=1}^{N} O_i(v) \geq \tau \right\}$$

In our implementation, we set the threshold $\tau = 0.9$.

3.4. Plane Estimation

As mentioned, we assume that the object is placed on a planar surface. After obtaining the spatial representation of the scene and the target object, we aim to estimate the plane equation of the planar surface, as the fitted plane will be used for obtaining the NSA (Section 3.6) and during the Skip-Rendering (Section 3.5). To this end, we first adaptively sample points that are likely to belong to the plane, using the 2D masks for each frame. Given those points, we then cast rays towards them and march along the ray until the density exceeds a certain threshold. Finally, we extract the depth values for those points and perform a linear regression to obtain the plane equation. In the following, we discuss each step in more detail.

Let D_i be the depth map that corresponds to image I_i. We split each image into 9 rectangle regions based on the 2D object bounding box. We take the 4 regions that are above, below, on the left, and on the right as candidate point sampling regions, which are denoted as R_1, R_2, R_3, R_4 for each image. We first calculate the mean depth for each region R_j in image i as follows:

$$\overline{D}_{i,j} = \frac{1}{|R_j|} \sum_{(x,y) \in R_j} D_i(x, y)$$

where $|R_j|$ is the number of pixels in region R_j, and $D_i(x, y)$ is the depth value at pixel (x, y) in the depth map D_i. We select the region with the minimum mean depth

$$R_{\text{min}} = \underset{R_j}{\text{argmin}}\, \overline{D}_{i,j}$$

as the image split to sample from. This practice is based on our observation that regions with smaller depth values tend

to include a larger area of the surface where the target object rests, and usually involve fewer occlusions.

We further define a sub-region inside R_{min} near the object and randomly sample a set of 2D pixels $P_i = \{p_1, p_2, \ldots, p_n\}$ from them. For each pixel p_k in P_i, we obtain its corresponding 3D point q_i using its depth value rendered from the NeRF model. As a result, we obtain a 3D point sample set $Q_i = \{q_1, q_2, \ldots, q_n\}$ of image I_i. The final set of sample points for plane regression is simply the union of Q_i across all images:

$$Q = \bigcup_i Q_i$$

Huber-regression is applied on the set of 3D points Q to find the best-fit plane, where we fix $d = -1$ in the plane equation $ax + by + cz + d = 0$ and solve for the parameters a, b, and c.

3.5. Skip Rendering

Within a specific image, there exist parts of the scene occluded by the target object yet visible from other perspectives. We perform skip rendering to avoid unnecessary inpainting, thus keeping our results more faithful to the real appearance. Based on the occupancy grids above the estimated plane, We construct a bounding box parallel to that plane, which will be used for skip rendering.

In volumetric rendering, transmittance is calculated by accumulating the spatial density along the ray propagation. Specifically, a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is emitted from the camera's center of projection o along the direction d.

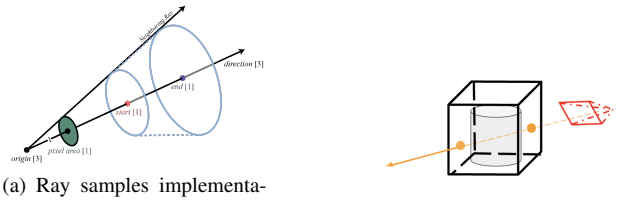
We modify nerfstudio [17]'s implementation of Ray Marching to allow skipping the target object in the rendering results. In nerfstudio, samples along a ray are represented as frustums (Fig. 3a) between the camera's predefined near and far planes t_n and t_f. As we assume that there is nothing in between the object and the camera, we can adjust the near plane to be the second intersection of the bounding box and the ray t_{b_k}. For each distance $t_k \in (t_{b_k}, t_f)$, the positional encoding of each ray position $\gamma(\mathbf{r}(t_k))$ is provided as input to an MLP parameterized by weights Θ, which outputs a density τ and an RGB color $\mathbf{c}$. In this way, the positional encoding features inside the bounding box will not be sampled, thus enabling skip rendering objects:

$$\forall t_k \in (t_{b_k}, t_f), \quad [\tau_k, \mathbf{c}_k] = \text{MLP}\left(\gamma\left(\mathbf{r}\left(t_k\right)\right); \Theta\right)$$

An example of occupancy grid creation, plane estimation and bounding box calculation can be found in Fig. 4.

3.6. Never-Seen Area

NeRF contains sufficient multi-view spatial information to fill in areas that are visible from at least a few, but not all, viewing angles, thus minimizing the area we need to inpaint. However, the piece of surface that the target object rests on is typically never visible in call frames. We define such regions as Never-Seen Area (NSA), which are the minimum area regions we need to perform inpainting on. NSA can be estimated by computing the intersection of the 3D object bounding box (Sec. 3.5) and the plane estimate (Sec. 3.4).

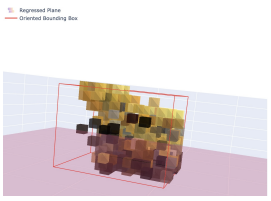


(a) Ray samples implementation in nerfstudio

(b) Skip the bounding box

Figure 3. **Demonstration of Skip Rendering**

(a) Original scene



(b) Estimated plane, bounding box, and occupancy grid representation of scene

Figure 4. **Example of occupancy grid creation, plane estimation, and bounding box computation**. Data is obtained from [14]

3.7. Inpainting and Geometric Consistency

The previous steps would result in 3D coordinates of the vertices of the NSA polygon. Using camera extrinsic and intrinsic parameters collected in Sec. 3.1, The polygon representation of the NSA can be projected back to any 2D image in the dataset. Without loss of generality, we choose the first image in the dataset and the associated 2D NSA as input for inpainting. Our inpainting method of choice is [16], identical to what was used in [20] and [10], with its set of default parameters. Once the NSA in one frame is inpainted, it can be warped to the rest of the frames by homography, again using the known intrinsic and extrinsic camera parameters, followed by Poisson image blending as described in [13] to keep the warped area consistent with the rest of the frame in terms of illumination and color.

3.8. NSA Refinement

Using the NSA-inpainted rendering images as input, we train the NeRF model for a second round and apply skip-rendering of the bounding box again to get the final scene representation where the object is completely erased while

having the NSA restored.

4. Results

4.1. User Interface

In order to provide the algorithm as an easy-to-use package for end users, akin to how Google's Magic Eraser works, we built a prototype web application that guides an end-user through the process of data capture, processing and presentation, while abstracting away the intermediate training steps. We built our application with Streamlit [2], for its ability to interface with the steps of our algorithm in Python code. A typical sequence of interaction with the application can be found in Fig. 10, and the corresponding screen recording can be accessed via this URL[1].

Since there are already excellent mobile applications such as Polycam for capturing RGBD data, we opted to delegate data collection to those purpose-built applications, while providing instructions to help with the successful capture of high-quality data, and requesting the user to export data from Polycam as a compressed ZIP file, which our training backend, NeRFStudio [17], can recognize and process.

Our application then presents one of the captured frames and asks the user for input by clicking on the object to be erased in the frame. The pixel coordinates of the object is then passed to the method described in Sec. 3.1. The mask generation algorithm occasionally yields imperfect results, therefore we present the generated mask as a colored overlay on the original frame for all frames in the dataset and provide the user with a slider to easily browse through the entire dataset to inspect the quality of 2D mask segmentation visually. In case of an imperfect mask, the user has the option to re-run the mask generation process by providing a different set of inputs.

Once the user has determined that the generated masks are correct, the captured data and generated mask can be used for training. Because of the high computational demands required by even the most efficient NeRF implementations [11], we opted for a server-client model. The backend runs on a dedicated PC with a Nvidia Graphics card and Nvidia's CUDA Toolkit, while the frontend application acts as a thin client that passes the validated training data to the backend. Steps outlined in Sec. 3.2 through Sec. 3.8 generally do not require user input, and are therefore abstracted away from the user interface. The user is informed about the training duration and continuously notified about the training progress. Once all steps are complete, the user is presented with a viewer, which relies on NeRFStudio's viewer, to interact with the trained NeRF scene without the object.

[1] https://youtu.be/RYbM-wvbCTo

4.2. Rendering Results

We display the qualitative results of our method in Figure 7. A video of our render result can be accessed via this URL[2]. Our method demonstrates a proficiency in spatially and temporally coherent reconstruction of scenes. The generated depth images distinctly show that the spatial region above the plane, previously occupied by the object, is now entirely vacated.

4.3. Ablation Study

4.3.1 User Study

In order to provide quantitative results, we conducted a user study to ask users to assess our user interface as well as our rendering results in comparison with the baseline video inpainting method DSTT [7]. As the NeRF training takes time and cannot be done in real-time, we made a video illustrating the whole process of removing an object in NeRF training using our user interface.

The user study questionnaire contains 10 questions in Figure 5:

1. From 1 to 5, to what extent do you think our UI is intuitive and easy to use based on the demo video above? (a demo video is given)

2. Only looking at the inpainted rendering result of our method, can you tell where the object was before? (rendered video is given)

3. Only looking at the inpainted rendering result of the DSTT Video Inpainting, can you tell where the object was before? (rendered video is given)

4. From 1 to 5, how would you rate the quality of the restored area where the object is erased in our method? (rendered videos are given)

5. From 1 to 5, how would you rate the quality of the restored area where the object is erased in DSTT 2D video inpainting? (Please ignore the lagginess in the DSTT result, but only consider the quality) (rendered video is given)

6. From 1 to 5, to what extent do you think our 3D inpainting result is temporally consistent? (rendered videos are given)

7. From 1 to 5, to what extent do you think the DSTT 2D video inpainting result is temporally consistent? (rendered video is given)

8. From 1 to 5, to what extent do you think our 3D inpainting result is spatially consistent? (rendered videos are given)

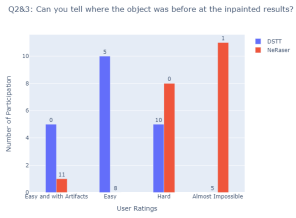
9. From 1 to 5, to what extent do you think the DSTT 2D video inpainting result is spatially consistent? (rendered video is given)

10. Do you have any suggestions or criticisms about our pipeline? (Open Question, optional)

Figure 5. **User study questionnaire questions**

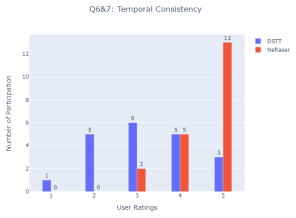
In particular, Questions No.2 and No.3 have four choices: It is almost impossible to tell even with careful inspection; It requires careful inspection to tell; It is easy to

[2] https://polybox.ethz.ch/index.php/s/4zfCVCvOD738515

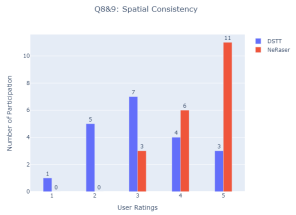


(a) Q2&3: original location of object

(b) Q4&5: Restoring Quality



(c) Q6&7: Temporal Consistency



(d) Q8&9: Spatial Consistency

Figure 6. User study responses

tell, but it looks reasonably natural; It is easy to tell with a rough look, and it looks totally unnatural. We label these answers with scores 4, 3, 2, and 1, respectively.

We calculated the average score of each question(except the open question 10) and used them as quantitative metrics for our method compared to the DSTT baseline. We collected in total 20 responses. Table 1 and Figure 6 show the results of the user study.

Question No.	1	2	3	4	5	6	7	8	9
Average Score↑	4.1	3	1.55	4.45	2.8	4.55	3.2	4.4	3.15

Table 1. Average scores of the user study responses.

From Table 1 we can see that our user interface scores 4.1/5, which reflects its user-friendliness. Figure 6 shows the responses to the paired questions comparing our method with the DSTT baseline. We can see that in all the questions our method received highly positive feedback. Many participants cannot tell where the object was even with careful inspection, and this proves that the inpainting quality of our method overwhelmingly surpasses that of DSTT. Moreover, the majority of participants assessed us with the highest score in questions No.6 and No.8. This strongly indicates that our approach delivers superior quality in both spatial and temporal consistency.

4.3.2 Effect of Plane Estimation

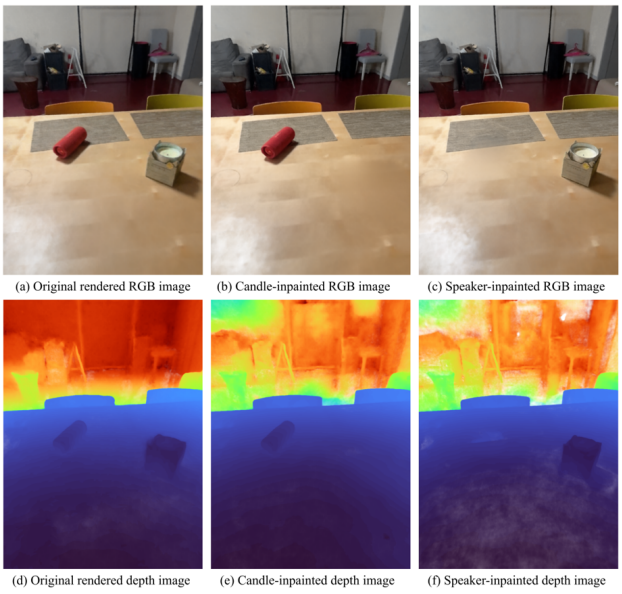
In Figure 8, we show a qualitative comparison of rendered images after skip rendering with and without plane estimation to demonstrate the effect of plane estimation. We apply plane estimation to rectify the skipped 3D bounding box to be strictly above the surface. As can be seen in Figure 8(a), the absence of plane estimation would cause a part of the resting surface to be undesirably skipped while rendering. With the rectified bounding box (Figure 8(b)), the NSA is well inpainted and there is no such artifact as seen in Figure 8(a).

4.3.3 Effect of Depth Supervision

In Figure 9, we show another qualitative comparison of sampled 3D points for plane estimation when the Nerfacto model is trained with or without depth supervision. We can see that in Figure 9(a) when the model is trained without ground truth depth maps, the sampled points tend to contain non-negligible noise. Furthermore, according to the relative position of the object (a can) and the estimated plane, the Nerfacto model also shows a bias of overestimating depth in the absence of ground truth depth maps. When the model is trained with depth supervision, as seen in Figure 9(b), such noise and bias are significantly suppressed and the estimated plane is accurate. Due to the detrimental impact that an inaccurate plane estimation would directly impose on downstream tasks such as homography warping of NSA's, we consider depth supervision a key component of the whole NeRaser pipeline to achieve high-quality inpainting.

5. Conclusion

In this report, we propose NeRaser, a novel method for object removal in NeRF-based 3D scene representations. To the best of our knowledge, NeRaser is the first to enforce both spatial and temporal consistency of inpainted areas among existing video inpainting and NeRF editing methods. We achieve such consistency through an elaborately designed pipeline including a novel skip rendering scheme and NSA estimation. We display qualitative rendering results from multiple self-captured scenes to demonstrate the effectiveness of our pipeline design. To make our method more accessible in a potential mixed-reality setting, we also built a prototype web application that guides an end-user through the entire NeRaser pipeline. Furthermore, we conducted a user study to help us evaluate the usability of the web application as well as the quality of our inpainting results against a baseline video inpainting method from an objective perspective. We hope NeRaser will inspire future innovations in flexible scene editing in mixed-reality environments.



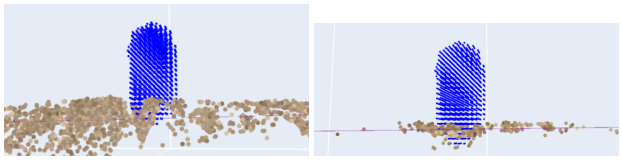
(a) Original rendered RGB image (b) Candle-inpainted RGB image (c) Speaker-inpainted RGB image

(d) Original rendered depth image (e) Candle-inpainted depth image (f) Speaker-inpainted depth image

Figure 7. **Rendering results of a scene with multiple objects to be removed.** Images (a), (b), and (c) are the rendered RGB from the original scene, the scene with the candle inpainted and the scene with the speaker inpainted, respectively. (d), (e), and (f) are their corresponding rendered depth images.

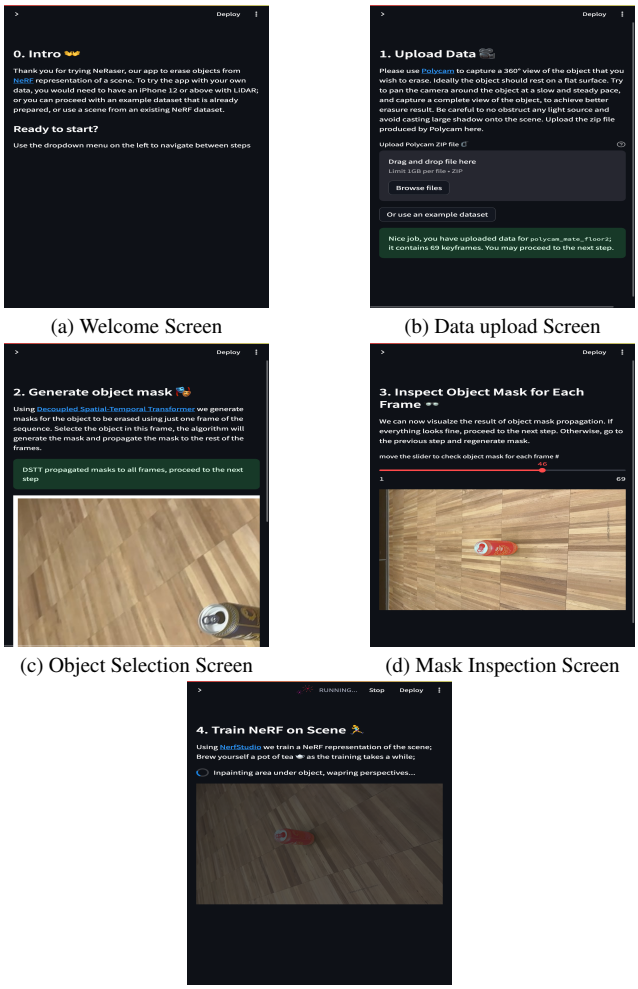
(a) Without plane estimation (b) With plane estimation

Figure 8. **Ablation study of plane estimation.** We show the rendered images after skip rendering both with and without plane estimation. We can see that (a) without plane estimation, there is still an artifact in the surface because of the unrectified bounding box. On the other hand, after rectifying the bounding box to be above the surface (b) with plane estimation, the NSA is well restored and there is no artifact left behind by the bounding box.



(a) Without depth supervision (b) With depth supervision

Figure 9. **Ablation study of depth supervision.** We show visualizations of the occupancy grid (blue points), sampled 3D points (colored by their corresponding sampled colors) and the estimated plane (purple). Between them, (a) is from a Nerfacto model trained without depth maps, while (b) is from another Nerfacto model trained with depth inputs. We can see that the presence of depth supervision suppresses the noise in sampled points significantly.



(a) Welcome Screen (b) Data upload Screen

(c) Object Selection Screen (d) Mask Inspection Screen

(e) Algorithm Training Screen

Figure 10. **Screenshots of user interface**, obtained from a typical interaction with the app on a mobile phone.

References

[1] Explore: Polycam." polycam - LiDAR & 3D scanner for iphone & android, poly.cam/explore. 2

[2] Streamlit • A faster way to build and share data apps — streamlit.io. https://streamlit.io/. [Accessed 01-01-2024]. 5

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

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

[5] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *NeurIPS*, 2021. 1, 2

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

[7] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenxiu Sun, Xiaogang Wang, and Li Hongsheng. Decoupled spatial-temporal transformer for video inpainting. *arXiv preprint arXiv:2104.06637*, 2021. 2, 5

[8] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[9] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 2

[10] 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. 4

[11] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. 2, 5

[12] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting, 2016. 2

[13] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, July 2003. 4

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

[15] Leila Shafarenko, Maria Petrou, and Josef Kittler. Automatic watershed segmentation of randomly textured color images. *IEEE transactions on Image Processing*, 6(11):1530–1544, 1997. 1

[16] 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. *arXiv preprint arXiv:2109.07161*, 2021. 3, 4

[17] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023. 2, 3, 4, 5

[18] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. *arXiv preprint arXiv:2305.15094*, 2023. 2

[19] Tengfei Wang, Hao Ouyang, and Qifeng Chen. Image inpainting with external-internal learning and monochromic bottleneck, 2021. 2

[20] 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. 2, 4

[21] Shangchen Zhou, Chongyi Li, Kelvin C.K Chan, and Chen Change Loy. ProPainter: Improving propagation and transformer for video inpainting. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2023. 2