Instruct-Object2Object: Object-Aware Editable Neural Radiance Fields With Prompts

Yuyang Tang*, Xutong Jiang*, Chenghao Gong*
* Department of Computer Science, University of Toronto

Abstract—Neural Radiance Fields (NeRF) represent a groundbreaking approach in computer vision, enabling the synthesis of high-fidelity 3D scenes from 2D images. This novel technique leverages neural networks to model volumetric scenes as a continuous function, mapping 3D coordinates to RGB colors and volumetric density values. Editable neural rendering aims to enable the editing of rendering results, such as user-defined object manipulation. This capability holds great application prospects in Augmented Reality (AR), Virtual Reality (VR), 3D design, and architecture. For this project, our goal is to design a user-friendly pipeline for object-aware neural rendering based on customized prompts. Using prompts, our model can segment objects through a zero-shot segmentation approach, allowing it to learn and render the object and the scene separately. This separation makes object displacement, such as moving or rotating within the scene, possible.

Index Terms—Computational Photography, Neural Radiance Fields, Deep Learning

1 Introduction

Recently, deep neural networks have been employed to learn novel view synthesis from 2D images. Methods like Neural Radiance Fields (NeRF) [1] represent scenes with implicit fields of volume density and view-dependent color and achieve photorealistic novel view synthesis results, where the scene and objects are static. Methods like Object-NeRF [2] propose object-compositional neural radiance fields that support the standalone object rendering. Moreover, due to occlusions, unseen parts of the original scene are usually degraded after editing. Nevertheless, this approach demands a substantial number of manually annotated 2D segmentation masks and preprocessed datasets, impacting the practicality and applicability of the method. Combined with diffusion models, methods like Instruct-NeRF2NeRF [3] allow for editing NeRF scenes with text instructions, where the model uses Instruct-Pixel2Pixel [4] to edit each image from the data sets as a whole without separating the object of interest and the scene, which means it only enables some color editing from diffusion model outputs, and sometimes not stable as the images are edited as a whole from one single prompt.

In this paper, we present a novel approach for object-aware neural rendering, which can be edited according to customized prompts. Our proposed pipeline leverages prompts to enable the model to perform zero-shot segmentation, allowing it to learn to edit and render objects and scenes separately. Our experiment shows that the contaminated scene limitation that exists in InstructNeRF2NeRF [3] is handled by local rendering on the masked object. The occlusion from editable objects is inpainted through texture synthesis methods, making the scene image set a smooth background with reasonable geometry, and prepared for further editing. This design facilitates object displacement, such as moving and rotating within the scene, providing a versatile and efficient solution for realistic and interactive scene and object manipulation.

2 Related Work

2.1 Neural Radiance Fields

Volume rendering allows the creation of a 2D projection of a 3D discretely sampled dataset. For a given camera position, a volume rendering algorithm obtains the R, G, B, α (Red, Green, Blue, and Alpha channel) for every voxel in the space through which rays from the camera are cast. The R, G, B, α values along the rays are then converted to an RGB color and recorded in the corresponding pixel of the 2D image. Neural Radiance Fields (NeRF) employ deep neural networks for an implicit representation of fields of volume density and view-dependent color and then accumulate the values in volume rendering, achieving realistic novel view synthesis results.

2.2 Object-Decomposite and Editable Rendering

Some early approaches such as [5] adopt traditional modeling and rendering pipelines to edit or insert objects in rendered scenes. Recently, some other methods have employed the NeRF model for 3D-aware image content manipulation. ControlNeRF [6] learns volumetric representations for multiple scenes simultaneously, and when a novel scene is required the rendering network is fixed and only the scene volume is optimized. In Object-NeRF [2], a dual-branch architecture is normally employed. The scene branch is trained to render the complete view of the scene and generate the background for editable scene rendering. Signed Distance Function (SDF) is also applied in neural rendering models such as VolSDF [7], ObjectSDF [8], to improve the rendering results of object surfaces. As the diffusion models emerge, Instruct-NeRF2NeRF uses the diffusion model Instruct-Pixel2Pixel [4] to enable prompt-driven NeRF editing. Instruct-Pixel2Pixel introduces a method for editing scenes using a combination of two large pre-trained models—a language model (GPT-3) and a text-to-image

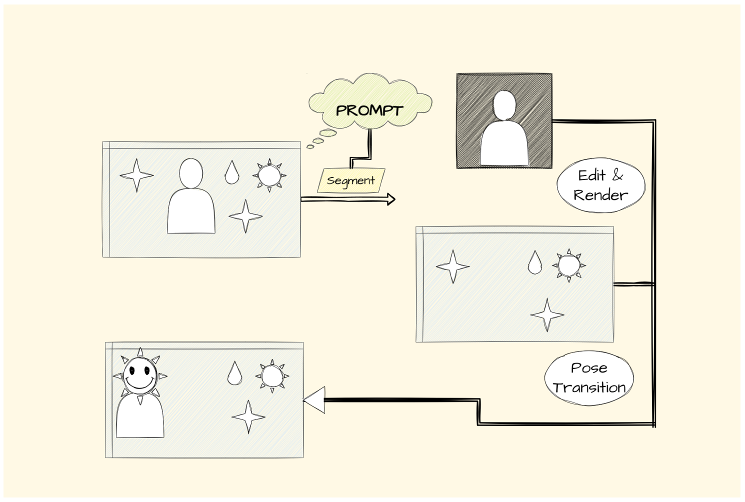

Fig. 1. General View of Proposed Editable Rendering

model (Stable Diffusion), which is utilized by Instruct-NeRF2NeRF to generate a replaced dataset of image editing examples, enabling holistic editing on rendering results of NeRF. Instruct-NeRF2NeRF maintains the geometry and position of objects in scenes by training the field model on raw images in the first stage. It only enables color editing in the second stage by learning from edited images from Instruct-Pixel2Pixel.

2.3 Image Inpainting

Several image inpainting methods have been developed, with traditional approaches such as texture synthesis capable of generating new textures similar to the surrounding areas to fill in missing regions. Deep learning neural techniques, including the use of Generative Adversarial Networks (GAN) and Context encoders, can be trained to generate realistic image patches as well. Neural methods have outperformed traditional methods, especially in semantic understanding of the image. However, neural models can be time-consuming to train and require a lot of data, thus computationally expensive, which is a trade-off we must consider when designing our model.

2.4 Segment anything model

The field of computer vision has witnessed significant advancements, particularly in the domain of image segmentation. Image segmentation, the task of partitioning an image into meaningful and semantically coherent regions, plays a pivotal role in numerous applications, ranging from medical imaging to autonomous vehicles.

The ”Segment Anything” model [9] stands out as a pioneering advancement, redefining the boundaries of semantic understanding within images. It consists of three components: an image encoder, a flexible prompt encoder, and a fast mask decoder. The image encoder is implemented using an MAE pre-trained vision transformer capable of processing high-resolution input. In this context, the prompt encoder is executed through CLIP (Contrastive Language-Image Pre-Training), a neural network trained with diverse (image, text) pairs. This system can be guided by natural language instructions to forecast the most pertinent text snippet associated with a given image [10]. Notably, it achieves this without direct optimization for the task, akin to the zero-shot capabilities demonstrated by GPT-2 and GPT-3.

By leveraging this pre-trained large language model on web-scale datasets, SAM demonstrates exceptional zero-shot and few-shot generalization capabilities. Traditional segmentation models often require extensive labeled data for each object class they aim to identify. In contrast, the ”Segment Anything” model exhibits a remarkable capacity to generalize to unseen classes with little to no explicit training, making it a powerful tool in scenarios where exhaustive annotation is impractical.

This ability enables faster preprocessing for our training dataset, as any objects can be segmented out from the background easily with a given prompt, making mask generation easier than ever.

3 Proposed Method

3.1 SAM-based data preprocessing

Compared with traditional NeRF training, our model requires additional processing on the training dataset. Our framework consists of two branches: the scene branch, which aims to encode the scene geometry and appearance, rendering the surrounding background; and the object branch, which renders the selected objects. The scene branch also assists the object branch in locating regions where occlusion occurs. Therefore, promptable SAM is used during the initial data processing stage, generating multiple binary masks as guidance.

Applying these input masks to the original image frame separates the scene and objects. These segmented components can then be fed into the scene branch and object branch separately for downstream training tasks. This approach enhances the model’s ability to capture nuanced scene features and object details during training, contributing to improved overall performance.

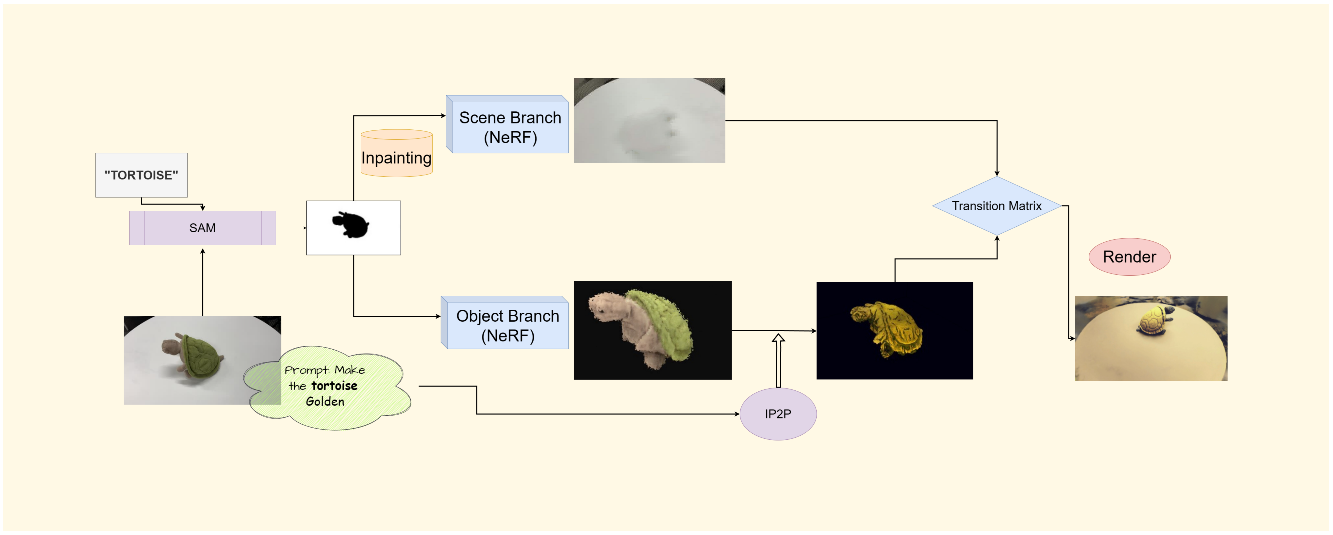

Fig. 2. Example Pipeline

3.2 Texture Synthesis

We decided to use the traditional approach for inpainting the part of images that were left empty because of the segmentation of objects for several reasons. As illustrated in the previous section, though neural methods can outperform traditional methods in restoring the original images, they also come with the disadvantage of consuming huge amounts of data and being computationally heavy during training. Also, current popular pre-trained models employing GAN [11] are mostly trained on the Places [12] dataset, which contains mainly real-world outdoor scenes, thus does not yield satisfying results as the scene in our project is just a single table within a relatively fixed indoor background. It is true that a new model could be trained using a related dataset of our scene, but that would add to the already heavy computational task our pipeline has involved, for which the current setting of time and resources of the project does not approve. Another reason we prefer the traditional approach is determined by the characteristics of our scene. Basically, we need to inpaint the segmented part of the white table from multiple directions of viewing, as well as a little bit of overlap between objects and the background. The task is not as complicated as inpainting outdoor scenes like street views and places of interest with crowds of people. This simplicity makes traditional methods more effective compared to the costly neural models. We are using the Rust-implemented version of fast texture synthesis([13]). The algorithm involved can be summarized as follow:

```
Algorithm 1: TextureSynthesis(I_a, OutputSize)
1 I_s ← $Initialize(OutputSize)$;
2 G_a ← $BuildPyramid(I_a)$;
3 G_s ← $BuildPyramid(I_s)$;
4 for Level L from lower to higher resolutions of $G_s$ do
5     loop through all pixels(x,y) of $G_s(L)$
6     $G_s(L)(x,y) = \_FindBestNeighbourhood(G_a, G_s, L, x, y)$
7 return ReconstructedPyramid($G_s$)
```

I_a is the original image, and Pyramid contains images of different resolutions, where lower level pixels neighbourhoods can also be used to represent higher resolution pixels during $FindBestNeighbourhood$ without sacrificing synthesis qualities.

3.3 Two-Branch Fields

We implemented two branches of NeRFs F_j and F_s to learn the colors and geometries of objects of interest and scenes respectively. In our first stage, the two fields are trained on separate data sets D_j and D_s, which have been preprocessed by part 3.1 and part 3.2 and have not been edited yet. As the occlusion part of the scene has been processed by the inpainting module, F_s will learn updated colors as well as geometries. Thus the fields output two sets of RGB and opacity values along the ray, which are accumulated when volume rendering, the object branch color and opacity outputs can be denoted as:

$$C_j(\mathbf{r_j}) = \sum_{i=1}^{N} T_{j,i}\alpha_{j,i}F_O(\mathbf{r_{j,i}}) \tag{1}$$

$$O_j(\mathbf{r_j}) = \sum_{i=1}^{N} T_{j,i}\alpha_{j,i} \tag{2}$$

$$T_j = exp(-\sum_{k=1}^{i-1} \sigma_{j,k}\delta_k) \tag{3}$$

where $\alpha_{j,i} = 1 - exp(-\sigma j, i\delta_i)$, and δ_i is the sampling distance between adjacent points along the ray. The scene branch is rendered in similar formulas from the field, supervised by inpainted ground truth. Considering the differences of distribution between objects of interest and the scene, i.e. the objects are usually closer to the observer, while the scene covers a larger range of distances from the viewpoint, we adopt different initial ray sample strategies for object branch and scene branch respectively:

$$\mathbf{r_j} = S_{UL}(\mathbf{ray}, d) \tag{4}$$

$$\mathbf{r_s} = S_U(\mathbf{ray}, d) \tag{5}$$

where S_{UL} is a sampler that allocates the first half of the samples uniformly and uses a linear disparity sampler in the second half, and S_U is a uniform sampler. $\mathbf{ray}$ denotes the initialized ray bundles and d is density set to sample. In this way, our model generates different point samples along the ray when initialized to fit the features of the objects and scenes.

3.4 Prompt-Driven Editing

In the second stage, like Instruct-NeRF2NeRF, the diffusion model Instruct-Pixel2Pixel is loaded to edit rendered images of the objects. The diffusion model receives three inputs: the original image from the training set (segmented objects) as the conditioning image C_I, the text prompt for object editing C_T, and the noised rendered image z_t. t represents the noise level and is chosen randomly from a fixed range $[t_{min}, t_{max}]$. As in SDEdit [14], the noised image z_t is then passed to a U-Net [15], where following Instruct-NeRF2NeRF the model only partially noises the images, and while the edited image is from rendered results, the diffusion model is conditioned on original images from the data set. The outputs of the diffusion model replace the training set after editing and are subsequently used to supervise the object-branch NeRF. The above process is operated iteratively during the training process, and over time as images are used to update the NeRF and progressively re-rendered and updated, outputs of object-branch NeRF begin to converge on a globally consistent depiction of the edited scene.

3.5 Loss Function

In the training process, the two branches are supervised simultaneously. In both branches Learned Perceptual Image Patch Similarity (LPIPS) [16] computes the similarity between the activations of two image patches for some predefined network, which has been shown to match human perception well. So besides the color L_1 loss, LPIP is employed to supervise the rendered images:

$$\mathcal{L}_c = \sum_{i \in M_{obj}} \|C_j(\mathbf{r_i}) - C_{j,gt}(\mathbf{r_i})\|_1 + \sum_{i \in I} \|C_j(\mathbf{r_i}) - C_{j,gt}(\mathbf{r_i})\|_1 \tag{6}$$

$$\mathcal{L}_{lpips} = d(\mathbf{I}_j, \mathbf{I}_{j,gt}) + d(\mathbf{I}_s, \mathbf{I}_{s,gt}) \tag{7}$$

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_{lpips} \tag{8}$$

When calculating the RGB loss of the object, the pixels of the scene are masked out, while for LPIPS loss they are set as zero (black). During the iterative editing stage, $C_{j,gt}$ and $\mathbf{I}_{j,gt}$ are replaced by the edited images such that the object-branch field learns to render edited images.

3.6 Editable Rendering

Object-NeRF [8] uses bounding boxes to avoid ray sampling in object spaces in the scene branch. However, in our model, the recognition of objects is from customized prompts, such that the 3D information of the objects like bounding boxes is unavailable. Therefore, in the object-branch field, we learn images where the background is masked as black. We obtain mask information from its rendering results and mask out the corresponding part of scene images during rendering. This approach, however, fails to deal with the situation in the scene occludes the objects, which is an important limitation. With trained two fields, transition matrix $\mathbf{T}$ can be applied to enable movement of the object (i.e. moving, rotating) in the scene:

$$C_{render} = \sum_{i \in M_{obj}} C_j(\mathbf{T}\mathbf{r_i}) + \sum_{i \notin M_{obj}} C_s(\mathbf{r_i}) \tag{9}$$

4 Experimental Results

4.1 Data

We trained and tested our method on three datasets directly generated from three casual videos, named Tortoise-Desk, Apple-Desk, and Sheep-Desk. In the NeRFStudio environment, the videos are firstly pre-processed by COLMAP [17] to match camera poses and generate sparse point clouds. With customized prompts, a prompt-driven Segment Anything model receives the images as inputs to generate relevant object masks. Due to the large computational costs in the editing stage, the images are down-scaled into the size of (270, 480).

4.2 Rendering Results

In Figure 4-6, we present the outcomes obtained from test sets across three datasets, encompassing three distinct scenarios: no changes made to the images, editing on the object of interest, and editing on both the object and the scene. Due to the limit of computational resources, our model only rendered and edited low-resolution images (270×480). Figure 4 showcases the reconstruction capability of our model, with noticeable jagged edges on the apple attributed to the SAM model's output masks exhibiting noisy edges The segmentation results of the Sheep-Desk data set are also not ideal and severely affect the performance of the object branch field. The masks generated by SAM model are sometimes not continuous, resulting in noises on the scene. This result shows that without any manual process, zero-shot segmentation models can hardly predict pixel-wisely precise masks.

In Figure 6, an attempt to simultaneously edit the background scene and object is depicted. However, Instruct-Pixel2Pixel is not tailored for whole image editing, causing suboptimal results when transforming the scene into diverse styles. For instance, when instructing the desk to transform to grass, the desk itself turns green, but the grass maintains a table-like structure with rounded edges. This limitation is inherent in Instruct-Pixel2Pixel as it struggles to alter background geometric structures.

Our model successfully avoids the contamination observed in diffusion models affecting backgrounds, a common issue with the Instruct-NeRF2NeRF model. In Figure 7, instances of the desk and other background elements turning orange are highlighted, resulting from the diffusion model erroneously editing the entire image at certain stages. Our approach, which involves rendering and editing the object and scene separately, significantly mitigates the likelihood of such contamination occurrences.

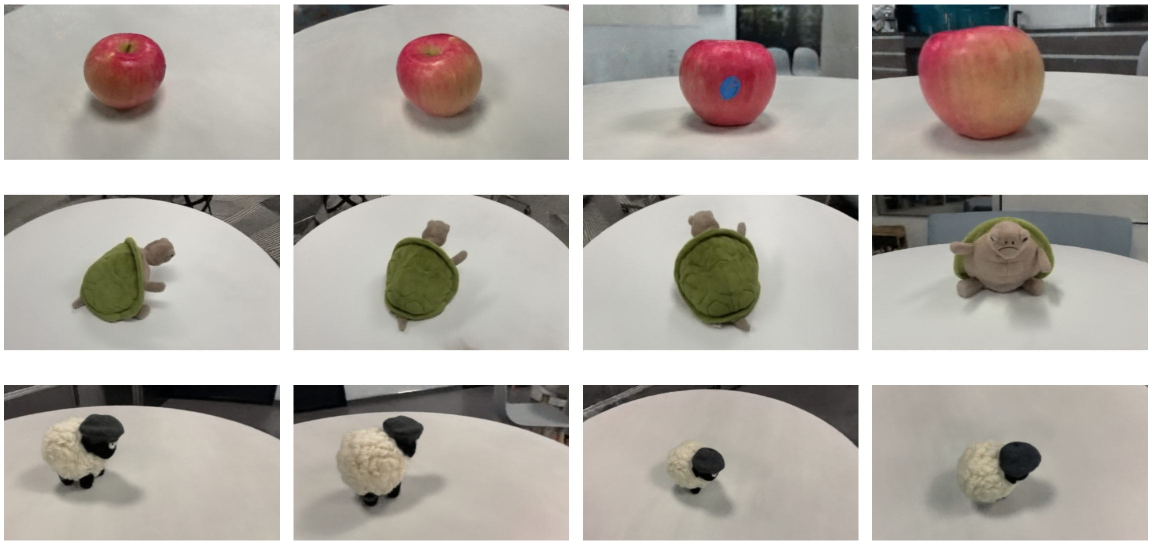
Fig. 3. Raw NeRF

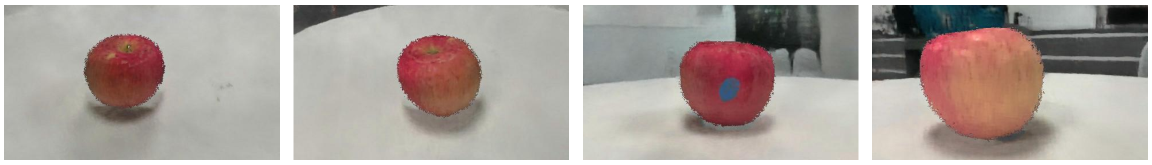
Fig. 4. No-Editing Results of Our Methods

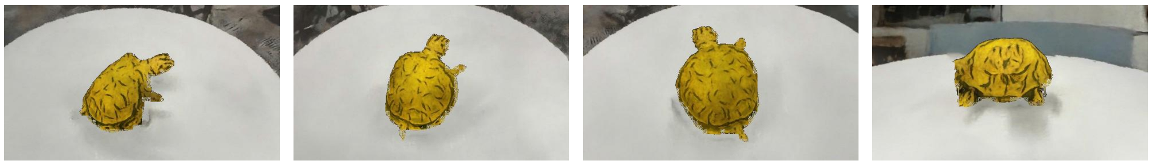
Fig. 5. Edited Results of Our Methods, with prompt ”make the tortoise golden”

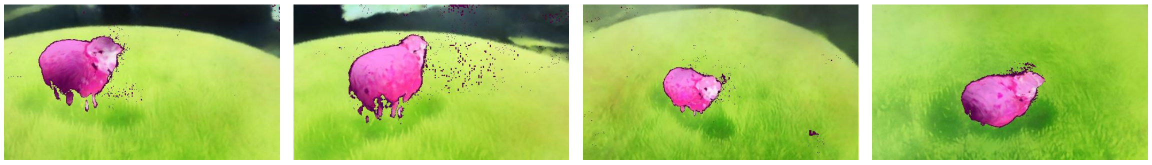
Fig. 6. Edited Results of Our Methods, with the prompt for the object ”make the sheep pink” and the prompt for scene ”turn the scene into a grassland”

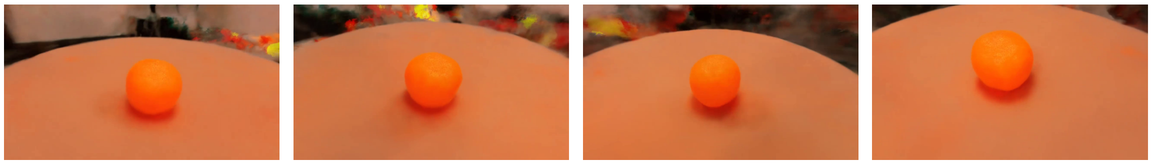
Fig. 7. Edited Results of Instruct-NeRF2NeRF, with the prompt for the object ”change the apple into an orange”

4.3 Quantative Results

Table 1-2 shows the quantitative results of comparisons between the original baseline NeRF and our model. The PSNRs or SSIM on objects denote the evaluation of only masked parts of objects. However, considering some noisy masks generated, the higher value in the object part represents over-fitting on incorrect masks (as the rendering results are evidently defective). Nevertheless, according to the evaluation of the full scene, our method shows reasonable compromises in performance when enabling object-aware editing.

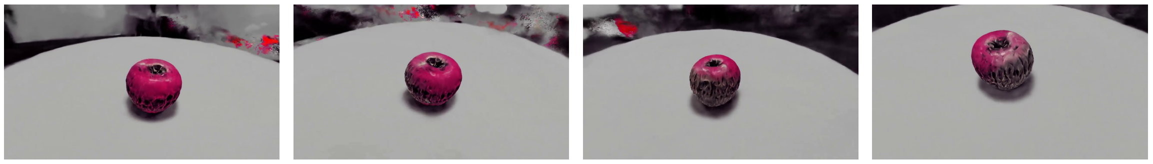

Fig. 8. Edited Results of Instruct-NeRF2NeRF, with the prompt for the object ”change the apple into a skull”

TABLE 1
PSNRs on object/full scenes/db

Model	Apple	Turtoise	Sheep
Raw NeRF (object)	25.87	27.67	27.66
IO2O (object)	25.87	25.86	25.86
Raw NeRF (full scene)	29.78	30.58	29.63
IO2O (full scene)	30.54	31.15	31.19

TABLE 2
SSIMs on object/full scenes

Model/	Apple	Turtoise	Sheep
Raw NeRF (object)	0.1211	0.0926	0.0506
IO2O (object)	0.1589	0.1148	0.0793
Raw NeRF (full scene)	0.9054	0.9129	0.9079
IO2O (full scene)	0.8521	0.8061	0.8298

5 Conclusion

5.1 Summary

In this paper, we propose a prompt-driven object-aware editing NeRF model. We combine the features of Object-Decompositional NeRFs and the work of Instruct-NeRF2NeRF and avoid contaminants on scenes when applying the diffusion model, as well as enabling editing on object positions such as rotating and moving.

5.2 Limitation and Future Work

The rendering results of our model are sometimes not ideal, especially on the edges of objects. As mentioned before, the performance of segmentation part significantly influences the supervision of the two branches of fields. Our next work will focus on automatic optimization of masks to filter out noises of masks. Furthermore, as mentioned in Section 3, our model can not handle the situation that the objects are occlude by the scene.

Our model also has the defect of being unable to handle symmetric objects. As shown in figure 8, Intruct-Pixel2Pixel tends to edit every image of the apple into the front of a skull, making geometry or color alignment impossible for the NeRF model. One possible way to solve this is to model the object in 3D meshes and apply another module to plan for the structure of edited object.

For image inpainting, we observed that while the algorithm generally performed well in texture synthesis for the hollows left by objects on the table and background, it struggled with handling shadows cast by the original objects. The root cause of this issue lies in the segmentation process, where only the objects themselves were removed from the scene, leaving their shadows behind. Consequently, when our algorithm seeks the optimal neighborhood for inpainting, it may kept selecting shadowed regions, resulting in a larger area of black pixels than intended. A potential future work involves training a neural GAN-based model using image data from similar scenes. Such a model is anticipated to make more informed decisions when encountering shadows compared to the traditional methods we have proposed, albeit at a higher computational cost.

Acknowledgments

The authors would like to thank Dr. David Lindell for his guidance and support throughout CSC2529: Computational Imaging at the University of Toronto. The authors are also grateful to the Teaching Assistants Anagh Malik and Parsa Mirdehghan for their valuable ideas and suggestions from the very beginning of this paper.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934
[2] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, “Learning object-compositional neural radiance field for editable scene rendering,” *CoRR*, vol. abs/2109.01847, 2021. [Online]. Available: https://arxiv.org/abs/2109.01847
[3] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa, “Instruct-nerf2nerf: Editing 3d scenes with instructions,” 2023.
[4] T. Brooks, A. Holynski, and A. A. Efros, “Instructpix2pix: Learning to follow image editing instructions,” 2023.
[5] K. Karsch, V. Hedau, D. A. Forsyth, and D. Hoiem, “Rendering synthetic objects into legacy photographs,” *CoRR*, vol. abs/1912.11565, 2019. [Online]. Available: http://arxiv.org/abs/1912.11565
[6] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll, “Control-nerf: Editable feature volumes for scene rendering and manipulation,” *arXiv preprint arXiv:2204.10850*, 2022.
[7] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, “Volume rendering of neural implicit surfaces,” *CoRR*, vol. abs/2106.12052, 2021. [Online]. Available: https://arxiv.org/abs/2106.12052
[8] Q. Wu, X. Liu, Y. Chen, K. Li, C. Zheng, J. Cai, and J. Zheng, “Object-compositional neural implicit surfaces,” 2022.
[9] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, “Segment anything,” 2023.
[10] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” 2021.
[11] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative image inpainting with contextual attention,” 2018.
[12] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[13] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 479–488.
[14] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," 2022.
[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597
[16] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018.
[17] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.