Blended-NeRF: Zero-Shot Object Generation and Blending In Existing Neural Radiance Fields (Supplementary Material)

Ori Gordon
The Hebrew University
ori.gordon@mail.huji.ac.il

Omri Avrahami
The Hebrew University
omri.avrahami@mail.huji.ac.il

Dani Lischinski
The Hebrew University
danix@mail.huji.ac.il

A. Additional Examples

We provide additional examples for the applications in the main paper. In Figure 1 we display additional views for the object replacement comparison with Volumetric Disentanglement for 3D Scene Manipulation [2]. In Figure 2 we demonstrate new object insertion using several captions from COCO dataset [9]. In Figure 3 and Figure 4 we show more examples for object replacement, and in Figure 5 and Figure 6 we display more edits and views for texture conversion task on 360 scenes.

B. Implementation Details

In this section we provide additional implementation details.

B.1. ROI Specification Interface

To specify the ROI and use it to decompose the scene, we introduce a graphic interface that enables positioning an axis-aligned 3D box inside the scene. Given the 3D position of the center, as well as the axis dimensions, of the box, rendering of the scene is performed from the provided camera position using the original NeRF model F_θ^O. The edges of the 3D box are then projected onto the image plane using the camera matrix. To provide intuitive feedback regarding the location of the box in the scene, we utilize the depth map of the scene to remove parts of the box edges that are occluded by the scene. In this manner, the user is able to specify the ROI in a precise and intuitive way by moving the box and modifying its dimensions while being able to inspect the location from any point of view.

B.2. Pose Sampling

In each training step, we sample a camera pose from a range of distances and angles, depending on the scene type. In the blender and 360 scenes, we sample azimuth and elevation angles in the ranges: $\theta \in [-180°, 180°]$, $\phi \in [-90°, 15°]$. For the radius, we first calculate the initial distance according to eq. (10) and then randomly sample the radius around this value. In llff dataset [12] we sample the camera pose from a spiral curve as used in the original NeRF implementation [1] . The curve is randomly sampled from a range of distances and radii in each axis. After sampling a camera pose, we recenter its rays around the ROI by moving its center location according to the center of mass inside the ROI (tracked by exponential moving average during training), but allow with a probability $p \in [0, 1]$ (hyperparameter, set to 0.1 in our experiments) to recenter the rays to a different point inside the ROI, with the aim of obtaining more versatile objects and densities. Additionally, we set the near and far planes (n, f) according to the box location and size in order to be more concentrated around the ROI and get more sample points per ray in this area:

$$n = d - \frac{D}{2}, \quad f = d + D, \tag{1}$$

where d is the distance of the camera from the center of mass inside the box and D is the box diagonal length.

B.3. Hyperparameters

In our experiments we set the max transmittance of L_T, the max variance of L_D and the weights of the losses to: $\tau = 0.88$, $\rho = 0.2$, $\lambda_T = 0.25$, $\lambda_D = 4$. We use the same network architecture as in [11] and the same hyperparameters and learning rates. To guide our model, we use the CLIP B/32 architecture.

B.4. Training

We train our model with a random seed value of 123 for all of our experiments. In experiments, we render the views at 168x168 resolution and up-sample to 224x224 resolution before feeding them to CLIP [15]. In the Comparisons and a Ablation study sections, we train the generator for 40,000 iterations and for the other figures in the main paper, the views resolution and the number of iterations depends on the complexity of the synthesized object and hardware limitations. We train with 4×24 GB A5000 GPUs. Training takes between a few hours to one day. We find that the

[1] https://github.com/bmild/nerf

(a) "aspen tree" (b) "strawberry"

Figure 1: **Additional views for object replacement comparison.** Additional views for the object replacement comparison with Volumetric Disentanglement [2]. The first and second rows display [2] and our results accordingly.

"bouguet of wilted red roses on a table." "broccoli laying on on a plastic board." "red and blue fire hydrant." "snowboard standing in a snow bank." "zebra eating grass on the ground."

Figure 2: **Object Insertion.** Insertion of new objects from COCO dataset [9] into an empty region in fern llff scene. Each column shows two views of the same edited scene [12].

primary driver for runtime/hardware requirements are the view resolution and the size of ROI (which require rendering more points along each ray).

B.5. Directional Dependent prompts

As described in the main paper, each iteration we concatenate a text prompt to the input caption depending on the camera location in the scene. We use the direction prompts below depending on the location:

- ", top-down view"
- ", front view"
- ", side view"
- ", back view"

In forward-facing scenes like llff dataset [12] we use the first three captions.

C. Additional Experiments Details

In this section we provide additional information regrading the experiments from the main paper.

C.1. Metrics

In our quantitative evaluation we report four metrics: CLIP Direction Similarity, CLIP Direction Consistency, LPIPS and R-Precision.

(a) original scene.

(b) edited scene.

Figure 3: **Object Insertion in vasedeck 360 scene.** We used the text: "a photo of a purple, white and blue flowers petals on the ground" and eq. (5) with $\alpha = 3.5$ to generate the edit.

(a) original scene.

(b) "a pineapple."

Figure 4: **Object replacement in 360 pinecone scene.** We replace the original pinecone object with pineapple using our proposed object replacement method.

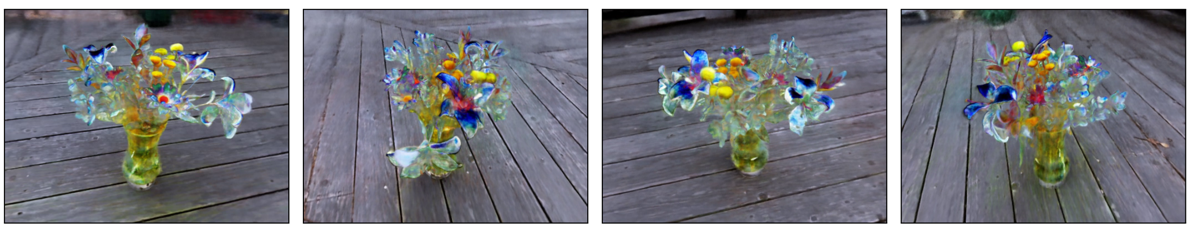
CLIP Direction Similarity introduced in [4] as a direction loss which measures the similarity between the change in the text descriptions and the change in the images. We use a variation of this metric so that high similarity will have high metric score:

$$\begin{aligned} \Delta T &= E_T(T_e) - E_T(T_o) \\ \Delta I &= E_I(I_e) - E_I(I_o) \\ L_{direction} &= \frac{\Delta T \cdot \Delta I}{|\Delta T||\Delta I|} \end{aligned} \tag{2}$$

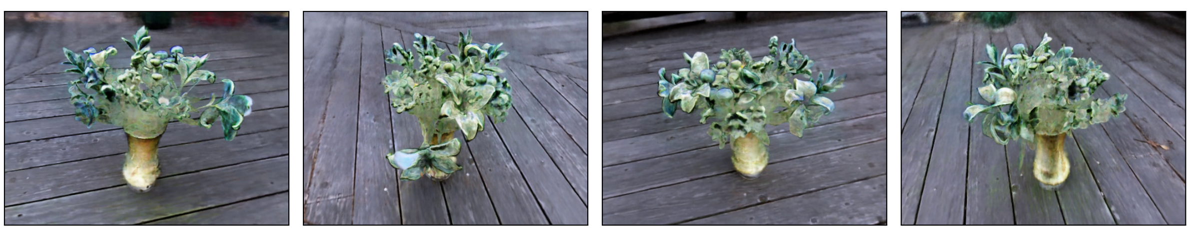
Original Scene. ”Burning pinecone”. ”Iced pinecone”. ”Golden pinecone”. ”Pinecone made of pink wool”.

Figure 5: **Texture conversion on 360 pinecone scene.**

(a) original Scene.



(b) “a vase made of glass.”



(c) “a vase made of stone.”

(d) “a water paint of a vase with flowers.”

Figure 6: **Texture conversion on 360 vasedeck scene.**

When E_T, E_I are the text and image encoders of CLIP, T_e, T_o are the text captions describing the edited and original scene inside the ROI and I_e, I_o are the according edited and original scenes views. In our experiments on the fern llff scene [12], we set T_o to: "a photo of a fern trunk".

CLIP Direction Score introduced in [5] measures the consistency between adjacent frames by calculating the CLIP embeddings of two corresponding pairs of consecutive views, one from the original scene and one from the edited scene. Similar to CLIP Direction Similarity metric, we then compute the similarity between the change in the original and edited scene views to get the final consistency score:

$$\begin{aligned} \Delta I_o &= E_I(I^o_{i+1}) - E_I(I^o_i) \\ \Delta I_e &= E_I(I^e_{i+1}) - E_I(I^e_i) \\ L_{direction} &= \frac{\Delta I_o \cdot \Delta I_e}{|\Delta I_o||\Delta I_e|} \end{aligned} \tag{3}$$

When I^o_i, I^o_{i+1} and I^e_i, I^e_{i+1} are the original and edited consecutive views pairs. In our experiments we compute this score on six consecutive views and average the results.

LPIPS or Learned Perceptual Image Patch Similarity, is used to judge the perceptual similarity between two images, [16] shows that this metric match human perception. The metric computes the similarity between the activation's of the two images for some network architecture. In our experiments we use LPIPS with pre-trained alexnet architecture [7] to measure the background similarity between the original and the edited scenes by masking the ROI region.

R-Precision [13] measures how well a rendered view of the synthesis object align with the text caption used to generate it. It computes the precision of the rendered views over a group of text captions using a retrieval model. Similar to DreamFields [6] we collect an object-centric captions dataset from COCO dataset [9] and sample 20 captions that will be used for training our model. We than compute the precision of the rendered views per synthesis object over the 153 captions. As the language image model backbone of the score, we use both CLIP [15] and BLIP2 [8], since we use CLIP to train our model.

D. Concurrent Work

Concurrently with our work, Instruct-NeRF2NeRF [5] present a diffusion-based method for editing a NeRF scene guided by text instructions. It utilizes InstructPix2Pix [3], which enables editing images based on text instructions. The edit is preformed by iteratively updating the image dataset of the original scene while training NeRF using these edited images. They demonstrate an impressive high quality local edit results on real scenes but sometimes can't preserve the rest of the scene and get a blurrier scene compared to the original, and sometimes even introduce texture and color changes to the entire scene.

SKED [10] research the possibility to edit a NeRF scene using guidance from 2D sketches from different views additional to an input text prompt describing the edit. They utilize the SDS loss presented in [14] to steer the edit towards the input caption and present preservation and silhouette priors to preserve the original scene and to preform the edit only on the sketched regions. In experiments they apply their method mainly on synthetic objects and demonstrate its applicability on objects insertion and replacement tasks such as hats, flowers and glasses.

In SINE [1], they suggest a method for editing NeRF scene by only editing a single view, and than apply the edit to the entire scene. To do this they encode the changes in geometry and texture over the original NeRF scene, by learning a prior-guided editing field. Using this field they render the modified object geometry and color and present color compositing layer supervised by the single edited view to apply the edit on novel views. They apply their method on real and synthetic scenes by changing the geometry and texture of objects in the scene.

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. *arXiv preprint arXiv:2303.13277*, 2023. 5

[2] Sagie Benaim, Frederik Warburg, Peter Ebert Christensen, and Serge Belongie. Volumetric disentanglement for 3d scene manipulation. *ArXiv*, abs/2206.02776, 2022. 1, 2

[3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022. 5

[4] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021. 3

[5] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023. 5

[6] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 5

[8] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. 5

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings ECCV 2014, Part V 13*, pages 740–755. Springer, 2014. 1, 2, 5

[10] Aryan Mikaeili, Or Perel, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. *arXiv preprint arXiv:2303.10735*, 2023. 5

[11] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1

[12] 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. 1, 2, 5

[13] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 5

[14] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 5

[15] 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*, 2021. 1, 5

[16] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5