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 $F^O_{\theta}$. 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 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:

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

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 $4 \times 24$ GB A5000 GPUs. Training takes between a few hours to one day. We find that the

1https://github.com/bmild/nerf
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 regarding 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.
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.

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:

$$\Delta T = E_T(T_e) - E_T(T_o)$$
$$\Delta I = E_I(I_e) - E_I(I_o)$$
$$L_{direction} = \frac{\Delta T \cdot \Delta I}{|\Delta T| |\Delta I|}$$

(2)
Figure 5: Texture conversion on 360 pinecone scene.

(a) Original Scene.  "Burning pinecone".  "Iced pinecone".  "Golden pinecone".  "Pinecone made of pink wool".

(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, $T_o$, $T_i$ are the text captions describing the edited and original scene inside the ROI and $I_o$, $I_i$ 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:

\[
\Delta I_o = E_I(I_o^{i+1}) - E_I(I_o^i)
\]
\[
\Delta I_e = E_I(I_e^{i+1}) - E_I(I_e^i)
\]
\[
L_{\text{direction}} = \frac{\Delta I_o \cdot \Delta I_e}{|\Delta I_o| |\Delta I_e|}
\]  

(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


[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


[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


