Language-driven Object Fusion into Neural Radiance Fields with Pose-Conditioned Dataset Updates – Supplementary Material –

Ka Chun Shum[1] Jaeyeon Kim[1] Binh-Son Hua[2,4] Duc Thanh Nguyen[3] Sai-Kit Yeung[1]

[1]Hong Kong University of Science and Technology [2]VinAI [3]Deakin University [4]Trinity College Dublin

Abstract

In this supplementary material, we provide more details of our proposed method with in-depth analysis and additional results. Specifically, we describe in detail the diffusion model used to blend objects into background images in an inpainting mode in Sec. 1. We present more ablation studies on additional aspects of our method in Sec. 2. We analyse the converge of NeRF training using pose-conditioned dataset updates in Sec. 3. We showcase the potential of our method via extensions in Sec. 4.

1. Diffusion model with inpainting mode

We detail the inpainting mode of the Stable Diffusion model [4] mentioned in Sec.3.2 in the main paper. Recall that D_θ (with parameters θ) is our diffusion model. D_θ includes a U-Net [5] denoising model ϵ_θ that learns the noise component $\epsilon_\alpha \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ added in the diffusion process, where $\alpha \in [0, 1]$ is the noise strength. To train D_θ with inpainting mode, a ground-truth inpainting dataset is required. Each data sample in such dataset is composed of an image I, a mask M, and an image-paired text prompt p.

Stable Diffusion [4] opts to accelerate the ϵ_θ training in a latent space rather than an image space. Specifically, a concatenated latent input for ϵ_θ is constructed as follows. First, given a training image I, a noisy latent vector $\mathcal{E}(I) \oplus \epsilon_\alpha$ is created, where $\mathcal{E}$ is an image encoder and $\oplus$ is a pixel-wise addition. The denoising model ϵ_θ is trained so that it can predict the added noise ϵ_α. Note that we skip writing tedious terms in the pixel-wise addition related to the forward/backward processes (please refer to DDPM [2] for mathematical details). Next, a masked image's latent vector $\mathcal{E}(I \odot M)$ is built, where $\odot$ is a pixel-wise product. $\mathcal{E}(I \odot M)$ feeds ground-truth mask-out information for ϵ_θ to perfectly restore the content outside the mask. Then, M is resized to a latent-shape mask M' to tell ϵ_θ the pixel locations that are inner-mask or outer-mask. Finally, the latent text prompt $\tau(p)$ encoded by a text encoder τ, and a diffusion time step t derived from ϵ_α (please refer to DDPM [2] for time step calculation), are passed to ϵ_θ as inputs. ϵ_θ is trained with the following loss:

$$\mathcal{L}(\theta) = \mathbb{E}\big[\|\epsilon_\alpha - \epsilon_\theta\big(\mathcal{E}(I) \oplus \epsilon_\alpha, \mathcal{E}(I \odot M), M', \tau(p), t\big)\|_2^2\big] \tag{1}$$

Note that, Eq. 1 here is a more detailed version of Eq. 1 in the main paper. However, for the sake of simplicity but without ambiguity, we skip subscripts representing variable distributions in $\mathbb{E}$.

2. More ablation studies

2.1. Quantitative results

In Sec. 4.6 and Fig. 5 in the main paper, we present qualitative results showing the impact of technical components in our method (e.g., diffusion model tuning with object/background images, dataset updates with pose-conditioned vs pose-random views, periodic dataset updates). Here we provide quantitative analyses of those components. In particular, we build different variants of our method, each of which corresponds a setting that we investigate in Sec. 4.6 in the main paper, e.g., we have variants which fine-tune the diffusion model using only object or background images. We report the CLIPScore and CLIPDC scores of our full pipeline and its variants in Tab. 1. Experimental results show that our full pipeline outperforms its variants on both the CLIPScore and CLIPDC metrics.

2.2. Noise strength hyperparameter

Recall that during NeRF updating, we pass NeRF rendering results to the fine-tuned diffusion models for pose-conditioned dataset updates. The diffusion model then refines the NeRF rendering results as in Eq. 6 in the main paper, where we set the noise strength $\alpha \in [0.3, 0.4]$. We found

Variant	CLIPScore ↑	CLIPDC ↑
Fine-tuning of the diffusion model without using object images	0.2861	0.1550
Fine-tuning of the diffusion model without using background images	0.3220	0.1891
Dataset updates with random poses (without pose-conditioned updates)	0.3216	0.1738
Without periodic dataset updates	0.3167	0.1887
Full pipeline (Ours)	**0.3276**	**0.1996**

Table 1. **Quantitative evaluations of technical components in the proposed method**. The results are complementary to those presented in Sec. 4.6 and Fig. 5 in the main paper. Higher scores indicate better and more consistent editing quality.

this setting effectively balances two objectives. First, it preserves sufficient color hints from previously learned nearby views (provided from the NeRF rendering results), thereby enabling the diffusion model to consider this color information. Second, it introduces an adequate level of randomness that enables the diffusion model to refine the NeRF rendering results based on the fine-tuned/pre-trained knowledge.

As this noise strength hyperparameter is critical for balancing the two objectives, we experiment it with different values and show corresponding results in Fig. 1. As shown, unsuitable noise strengths used by the diffusion model result in various failures. In particular, a too-low noise hinders the diffusion model from refining the NeRF rendering results with plausible object or background content, largely reducing the effectiveness of the diffusion model during inference (see Fig. 1-a). On the other hand, a too-strong noise disables the nearby-view color hints being passed to the diffusion model and hence defects the dataset updates strategy, resulting in view inconsistencies (see Fig. 1-c,d). We also test the diffusion model in inference with random strengths within a range $[0.02, 0.98]$ as in Instruct-NeRF2NeRF [1]. We found that high-noise inference dominates and leads to similar pose inconsistency (see Fig. 1-e). As shown in the results, our current setting (see Fig. 1-b) well balances the objectives and thus can make plausible content with view consistency.

3. Convergence analysis

In the context of pose-conditioned dataset updates, we progressively include new nearby views into the NeRF training dataset. In this study, we provide a detailed example of how the generated content in these newly included nearby views converges to a desirable quality. Specifically, we first train the NeRF on an initial view (1^{st} view) for 500 steps. We then include two nearby views (2^{nd} and 3^{rd} views) after rendering them by the NeRF and refining them by the diffusion model. These two newly included views are trained from step-500 to step-1000. We show in Fig. 2 the latest NeRF renderings and the diffusion model-refined images at step-500, step-750, and step-1000. The results reveal a gradual convergence of objects from defected to better quality (observe the zoom-in regions). These results clearly show the effectiveness of our pose-conditioned dataset update strategy.

It is worth noting that Fig. 2 also reflects two important observations that motivate our work. First, the NeRF is capable of rendering sufficient object color hints in new views learned from previous nearby views, even at step-500 where the new views are not yet seen by the NeRF. Second, artifacts can be addressed by our diffusion model, fine-tuned on object images. This fine-tuning constrains an object from diverging to different appearances or inconsistent poses.

4. Extensions

4.1. Object locating using bounding box

Our pipeline allows object locating using bounding boxes. We show several results of this experiment in Fig. 3, where we insert the same object into a background scene at different locations using both 3D bounding box (as in our current setting) and 2D bounding box. We observed that, 3D bounding boxes can fit most the real-world objects (see Fig. 3-a). On the other hand, 2D bounding boxes can locate flat objects on a surface but fail to describe objects which exhibit 3D-aware perspective (see Fig. 3-b).

4.2. Insertion of synthetic objects via text-to-3D

We show the versatility of our method by allowing synthetic objects to be inserted in real-world background scenes. This is enabled by leveraging the advanced text-to-3D technique in [3] to create 3D models, whose multi-view renderings are used as object images in our pipeline. We illustrate several results of this study in Fig. 4. Although the objects generated by the text-to-3D technique are synthetic, our method still produces view-consistent editing results. We hypothesize that future advances in text-to-3D generation will improve the texture and lighting conditions of the synthesized 3D models, making object fusion results into real-world background scenes more photo-realistic.

4.3. Multiple editings

Editing a scene with different operations for multiple times is a natural practice in scene editing. Our pipeline also supports this demand. Specifically, we can iteratively perform

(a) Noise strength $\alpha = 0.05$ (low noise strength)

(b) Noise strength $\alpha = 0.35$ (Ours, relatively low noise strength)

(c) Noise strength $\alpha = 0.65$ (relatively high noise strength)

(d) Noise strength $\alpha = 0.95$ (high noise strength)

(e) Noise strength α randomly sampled from range $[0.02, 0.98]$ [1] (random noise strength)

Figure 1. **Ablation study results about noise strength**. Each row shows the results of a variant of our pipeline adapting a different noise strength during training. The left and right columns include images of two different views of an edited scene. For each column, from left to right are the results of increasing training steps, where the most left image in each column is an early-stage result and the most right is the final output. The left column is a view near the starting view, which converges faster than the right column from a farther view.

multiple editings (one by one) through our pipeline on the same scene. We showcase this ability in Fig. 5, where we apply our method for an editing on a background scene. We then treat the edited scene as a new input background on which we apply our method again for another editing. The results in Fig. 5 illustrate the ability of our method in making multiple view-consistent editings.

References

[1] 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. 2, 3

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33:6840–6851, 2020. 1

[3] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988, 2022. 2, 5

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022. 1

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pages 234–241. Springer, 2015. 1

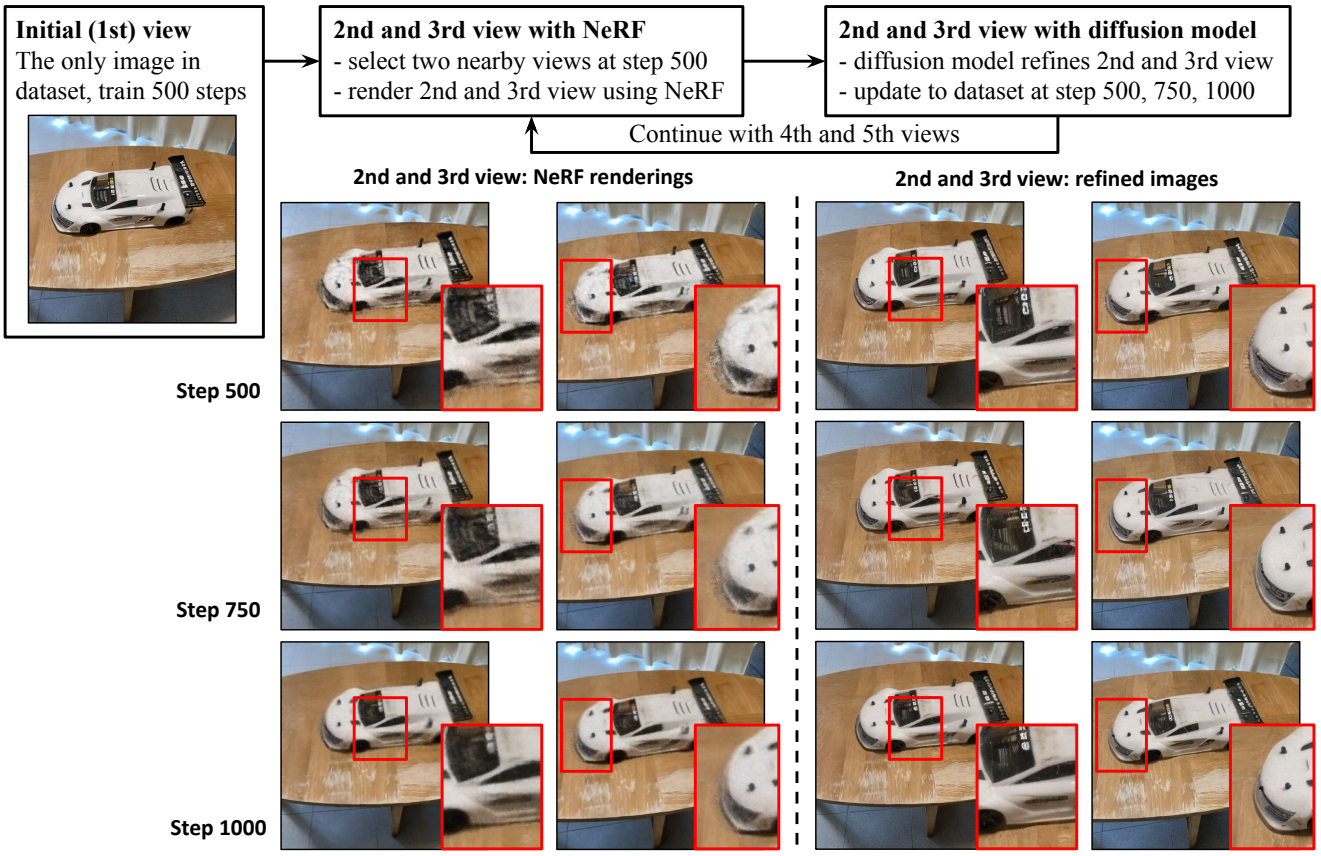
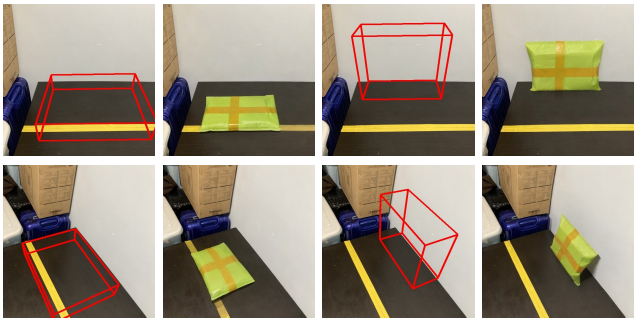
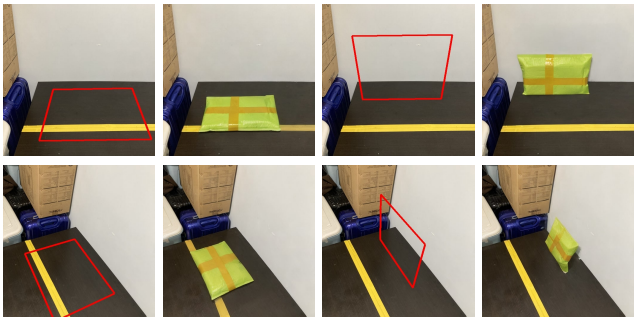


Figure 2. **Convergence of newly included near-by views**. We show a convergence example of the generated content in the newly included nearby views. Red boxes with corresponding closeups highlight the editing quality achieved by the joint 2D diffusion-3D NeRF updating.

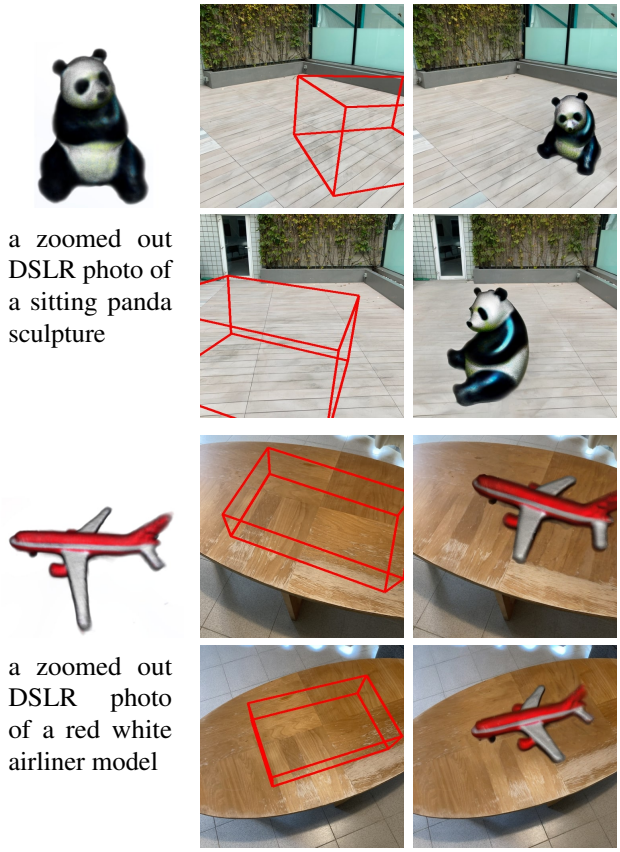


(a) 3D bounding boxes can effectively specify the object orientation



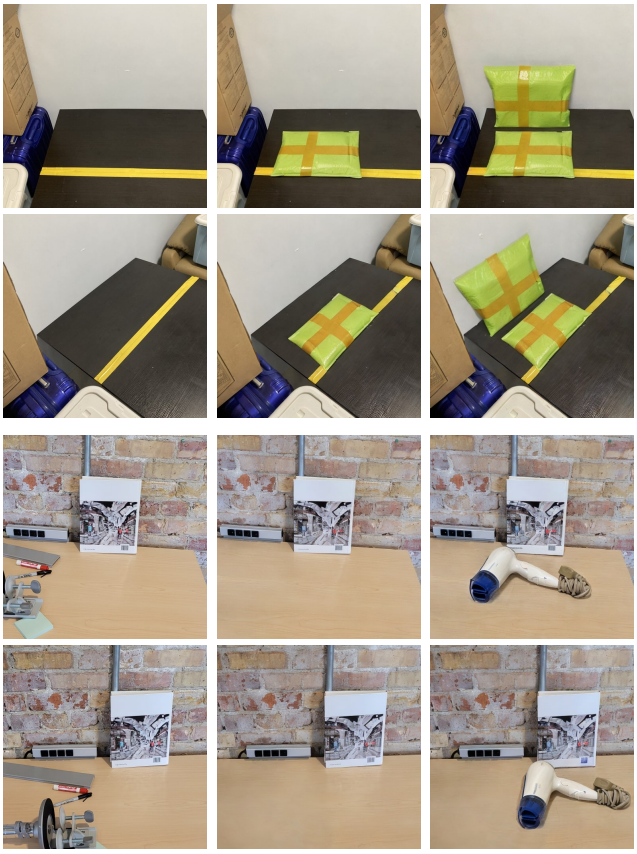
(b) 2D bounding box works for objects that fit a 2D surface in all viewpoints (left) but fails for objects viewed from more 3D-aware perspectives (right)

Figure 3. **Object locating using 3D vs. 2D bounding box.** Here we show the effect of object locating using 3D bounding box (a) and 2D bounding box (b). First, we specify the locations for a target object by placing a bounding box into a background scene (first and third columns). The object is then inserted by our method at corresponding locations (second and fourth columns). As shown, our current setting with 3D bounding box design maintains better view consistency and 3D awareness.



(a) Text-to-3D object and text prompt (b) Input background and target bounding box (c) Edited scene from our full pipeline

Figure 4. **Insertion of synthetic objects produced by text-to-3D generation**. (a) Text-to-3D objects generated by [3]. Multi-view images of these objects are then rendered for our pipeline. (b) Real-world background scene with bounding box providing location for the target object. (c) Editing results. As shown, our method can generate pose-consistent editing results even with synthetic-looking objects.



(a) Input (b) First edit (c) Second edit

Figure 5. **Multiple editings on the same scene**. (a) Input background scenes. (b) First editing: we insert a yellow package into the first scene (top two rows) and remove the accessories on the table from the second scene (bottom two rows). (c) Second editing: we insert another yellow package into the first scene (top two rows) and insert a hair dryer into the second scene (bottom two rows).