DATENeRF: Depth-Aware Text-based Editing of NeRFs

Sara Rojas[1⋆], Julien Philip[2], Kai Zhang[2], Sai Bi[2], Fujun Luan[2], Bernard Ghanem[1], and Kalyan Sunkavalli[2]

[1] KAUST
{sara.rojasmartinez,bernard.ghanem}@kaust.edu.sa
[2] Adobe Research
{juphilip,kaiz,sbi,fluan,sunkaval}@adobe.com
https://datenerf.github.io/DATENeRF/

Fig. 1: DATENeRF uses a reconstructed NeRF scenes's depth to guide text-based image edits. Compared to the state-of-the-art Instruct-NeRF2NeRF [13] method (top row), our method (bottom row) produces results that are significantly more photorealistic and better preserve high-frequency details across a diverse range of text prompts.

Abstract. Recent diffusion models have demonstrated impressive capabilities for text-based 2D image editing. Applying similar ideas to edit a NeRF scene [31] remains challenging as editing 2D frames individually does not produce multiview-consistent results. We make the key observation that the geometry of a NeRF scene provides a way to unify these 2D edits. We leverage this geometry in depth-conditioned ControlNet [57] to improve the consistency of individual 2D image edits. Furthermore, we propose an inpainting scheme that uses the NeRF scene depth to propagate 2D edits across images while staying robust to errors and resampling issues. We demonstrate that this leads to more consistent, realistic and detailed editing results compared to previous state-of-the-art text-based NeRF editing methods.

Keywords: 3D Scene Editing · Neural Rendering · Diffusion Models

⋆ Work done during an internship at Adobe Research.

1 Introduction

The recent progress in Neural Radiance Field (NeRF)-based methods [7, 31, 33] has now made it possible to reconstruct and render natural 3D environments with an ease and visual quality that has previously not been possible with traditional 3D representations. That said, traditional 3D representations like textured meshes explicitly decouple geometry and appearance; this gives artists, albeit with significant skill and time, the ability to make complex edits to 3D scenes and produce visually compelling results. This task becomes particularly challenging when dealing with NeRFs because they lack explicit representations of surfaces and appearances.

At the same time, image synthesis and editing have been revolutionized by 2D diffusion-based generative models [38, 42, 43]. These models can generate (or edit) images using text prompts, inpaint masked regions in images [1] or edit images following user instructions [5]. In cases where text prompts are not a fine-grained enough edit modality, approaches such as ControlNet [57] enable the generation and editing of content conditioned on spatial guidance signals, including but not limited to depth, edges, and segmentation maps.

Recent work has explored using such 2D diffusion models to edit 3D NeRF scenes [13, 48]. However, editing individual images of the same scene (with diffusion models or otherwise) produces inconsistent results that require different forms of regularization [30, 47] and/or relying on the NeRF optimization to resolve [13]. This is successful only up to a point; for example, as can be seen in Fig. 1 (top), even the state-of-the-art Instruct-NeRF2NeRF method [13] suffers from errors in geometry, blurry textures, and poor text alignment.

We address this challenge using DATENeRF, a Depth-Aware Text-Editing method that uses the reconstructed NeRF geometry to improve the consistency of individual 2D edits. We propose using ControlNet [57], conditioned on the NeRF depth, as the base 2D diffusion model for text editing. This depth conditioning improves the geometric alignment of edited images but they can still have very different appearance. To address this, we propose reprojecting edited pixels in one view onto the next view using the NeRF depth. Doing this naively produces poor results because errors in geometry and resampling issues aggregate over views. Instead, we use the reprojected pixels to initialize a hybrid inpainting step that inpaints disoccluded pixels but also refines the entire image to produces 2D images that are both high-quality and consistent.

This improved consistency means that the edited images can be easily fused by a subsequent NeRF optimization to produce a high-quality edited NeRF scene. As can be seen in Fig. 1 (bottom), DATENeRF produces results that have cleaner geometry and more detailed textures compared to Instruct-NeRF2NeRF which blurs these details out because of the inconsistencies in 2D edits. Moreover, by incorporating ControlNet into NeRF editing, we open up a broad spectrum of fine-grained NeRF modification capabilities, encompassing both edge-based scene alterations and the insertion of objects, as showcased in Fig. 7 and 8, respectively. This integration enhances the controlability of scene editing.

2 Related Work

NeRF Editing. While there has been extensive research, and even development of commercial software tools for editing 3D content, these have been traditionally applied to textured meshes or point clouds. The emergence of NeRF-based reconstruction methods [7, 31, 33, 41] made it easy to reconstruct 3D representations from 2D images, thus opening up the requirement for tools to edit these representations. The optimization-based approach for reconstructing NeRFs is also amenable to editing tasks. As a result, many methods have been proposed to edit a trained NeRF model by re-optimizing it based on shape/color scribbles [28], exemplar styles [10, 16, 17, 34, 47, 56], and changes to color palettes [18, 24, 53]. Other methods have proposed physically-based editing tools for NeRFs including compositing [51,52], deformations [20,35,55], object removal [32], relighting and material editing [4, 25, 59]. All these methods only allow for specific, low-level edits; in contrast, we propose a general text-based editing method for NeRFs.

3D Editing with vision-language and diffusion models. Powerful vision-language models like CLIP [37] have been used for NeRF generation and editing [12,19,46] and distilling CLIP features into 3D [21,23]. The high-level nature of the CLIP features means that these methods can only demonstrate coarse forms of edits unlike the fine-grained, visually higher quality edits we demonstrate. SINE [2] transfers edits from a single edited image across the entire scene using a ViT model [6] as a semantic texture prior.

There have also been incredible advances in text-based 2D generative diffusion models [38, 42, 43]. Methods have also been proposed to condition these generative models on additional control signals [57] and instructions [5]. Recent works have applied these approaches to 3D representations. 3D generative models have been proposed to generate NeRFs by using an SDS loss [36] from pre-trained 2D generators via optimization [9, 26, 36, 49, 50]. The SDS loss has also been used to edit NeRF models [44, 54]; however, the quality of the results is sub-optimal. DreamEditor [60] also uses the SDS loss to edit NeRFs but requiring finetuning the diffusion model on the input scene. In contrast, we use a pretrained diffusion model.

Our work builds on the state-of-the-art Instruct-NeRF2NeRF method [13] for text-based NeRF editing. This method proposes an "Iterative Dataset Update" approach which alternates between editing individual input images (that can lead to inconsistent results) and NeRF optimization (that resolves this inconsistency). However, this approach converges slowly, and struggles with high-frequency textures and detailed edits because of its inherent stochasticity. In contrast, we propose explicitly using the NeRF geometry to make the image edits consistent, thus leading to faster NeRF convergence and higher quality results. Similar to us, ViCA-NeRF [11] uses depth to enforce view consistency in the edits. However, it does so via blending of projected latent codes; this requires more passes of a diffusion model and leads to blurrier results than ours.

2D diffusion models have also been used for texturing traditional 3D representations like polygonal meshes [8, 40]. These methods project generated 2D

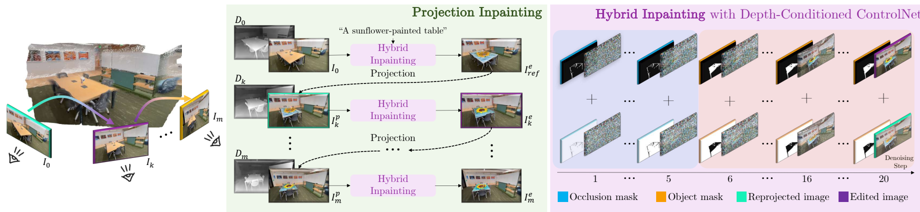

Fig. 2: Overview. Our input is a NeRF (with its posed input images) and per-view editing masks and an edit text prompt. We use the NeRF depth to condition the masked region inpainting. We reproject this edited result to a subsequent viewpoint and using a hybrid inpainting scheme that first only inpaints disoccluded regions and then refines the entire masked region. This is done by changing the inpainting masks (indicated by the blue and orange blocks on the right side) during diffusion.

images onto the 3D mesh, do this iteratively for a carefully selected set of viewpoints, and merge the generated images into a consistent texture space using the UV unwrapping of the given ground truth 3D mesh. Our method also projects generated 2D images onto the 3D NeRF space for editing but is designed to handle NeRF reconstructions from in-the-wild scene captures with potential errors in geometry, no texture unwrapping, and arbitrary input viewpoints.

3 Method

Given a set of input images $\{I_1, I_2, \ldots, I_m\}$ (with corresponding camera calibration), we reconstruct a 3D Neural Radiance Field (NeRF). This NeRF model represents the scene as a volumetric field with RGB color and volume density at each 3D location and enables rendering of novel views using volume rendering. Our goal is to allow users to edit specific regions of this NeRF scene, denoted by masks $\{M_1, M_2, \ldots, M_m\}$, using text prompts. We leverage the power of 2D diffusion models to make complex text-based edits to the constituent 2D images of the scene. Independently editing each 2D image leads to view inconsistencies that, when merged into an edited NeRF, produce results with blurry textures and geometry artifacts. We propose to use the scene depth reconstructed by NeRF to resolve these inconsistencies.

We choose ControlNet [57] conditioned on NeRF depth (Sec. 3.2) as our base 2D editing model. This ensures that the major features in the edited images are coarsely aligned with scene geometry and, as a result, more consistent across views. However, this by itself is not sufficient. We further leverage scene geometry by explicitly reprojecting edits made to an image to other views. We account for these reprojected pixels in the diffusion process via a projection inpainting step (Sec. 3.3) to improve view consistency as well as preserve visual quality. This results in consistent 2D images that can be fused into a high-quality edited NeRF scene via optimization. The full DATENeRF method is illustrated in Fig. 2.

3.1 3D-consistent region segmentation

We first specify how we compute the masks $\{M_k\}$ used for per-view editing. We can use volume rendering on the NeRF geometry to compute an expected distance per pixel for any given NeRF viewpoint; we denote these distance maps for the input viewpoints as $\{D_1, D_2, \dots, D_m\}$.

Given a target object to be edited, we generate initial per-view segmentation masks using an off-the-shelf segmentation model [22, 27]. These masks tend to have inaccuracies and are inconsistent with each other. To rectify these issues, we *aggregate* these masks in 3D using the NeRF scene geometry. Specifically, we unproject each pixel within each preliminary mask, M_k, into 3D points using the NeRF distances $\{D_k\}$. We project each of these points into all the masks $\{M_1, \dots M_m\}$ and assign a selection score that is accumulated from the initial per-view masks. Only those points that surpass a pre-defined visibility threshold are retained in an updated view-consistent point cloud. We remove outliers points that lie outside a specified sphere centered on the object. The refined points are then projected back into the input views to create updated masks. We finally employ a guided filter [14] to filter the masks (guided by the RGB images) to create the final masks. The result of this process is a set of clean, occlusion-aware masks that are view-consistent and are used for subsequent processing steps. With some abuse of notation, we refer to these final masks as $\{M_k\}$.

For details, please see supplementary. We now describe how we use the input images $\{I_k\}$, masks $\{M_k\}$ and NeRF geometry $\{D_k\}$ to edit the NeRF scene.

3.2 Editing NeRFs with Depth-aware ControlNet

Inpainting with 2D diffusion models. Diffusion models, especially Denoising Diffusion Probabilistic Models (DDPM) [15], have gained prominence in generative modeling. At their core, these models transform a normal distribution into a target distribution through a series of denoising steps. In this work, we use Stable Diffusion, which is a latent diffusion model [42].

A text-to-image model can be applied for inpainting by adjusting the diffusion steps to account for known regions [1, 29]; we specifically use Blended Diffusion [1]. Here, the denoising operation is applied to the full noised image latents at every step but the denoised result is replaced by the noised input latents in the regions outside the pre-defined inpainting mask. This ensures that the final result retains the original image outside the mask, but generates the masked regions that are consistent with the text prompt and the outside regions.

Incorporating ControlNet for image editing. As can be seen in Fig. 3 (row B), inpainting the mask regions of the input images using the method detailed above leads to a wide range of inconsistent changes across the images of the scene. Our goal is to reduce these inconsistencies. Toward this goal, we propose conditioning the image generation/inpainting on the scene geometry. We achieve this by converting the NeRF distances $\{D_k\}$ to per-view disparities and using them as conditioning for a ControlNet [57] model. Combining this with the Blended Diffusion step detailed above, we compute edited images as:

$$I_k^e = \text{Blended-Diffusion}(\text{ControlNet}(I_k, D_k), M_k). \tag{1}$$

As can be seen in Fig. 3 (row C), using ControlNet produces more spatially coherent and context-aware synthesized results. Note that Instruct-NeRF2NeRF [13] also relies on conditioning the editing process to improve view consistency. However, in their case they use the input images as conditioning. In contrast, we use depth as conditioning, thus making the model more flexible in its ability to produce content that could be significantly different from the input images. For more details, the pseudocode of this section can be found in the supplementary.

3.3 Projection Inpainting

As can be seen in Eqn. 1, up to this point, each image in the scene is edited independently and there is only a weak form of view consistency being enforced via the depth conditioning. Previous methods rely on NeRF optimization to iron out these deviations but this does not work especially for high-frequency content, where small misalignments in images can lead to blurry NeRF results.

We address this with a simple observation: relying on NeRF optimization to propagate edits across images is an indirect mechanism; instead, we explicitly leverage scene geometry to achieve this. Thus, given a single edited reference viewpoint, I_{ref}^e, we reproject the edited pixel values to other viewpoints to directly build a set of edited views that are consistent by construction:

$$I_k^p = R_{\text{ref}\to k}(I_{\text{ref}}^e), M_k^{\text{vis}} = R_{\text{ref}\to k}(M_{\text{ref}}). \tag{2}$$

Here M_k^{vis} denotes which regions of I_k^e are being reprojected from I_{ref}^e and are mutually visible in these two viewpoints (using a depth test, see supplementary). In practice, we project pixels from other views and resample the reference view.

These reprojected images already give us a sense of what the edited viewpoints should look like. Similar to how we used Blended Diffusion to inpaint only the edited regions, we can preserve the reprojected pixel values as:

$$I_k^e = \text{Blended-Diffusion}(\text{ControlNet}(I_k^p, D_k), M_k^p). \tag{3}$$

Here, $M_k^p = M_k * (1 - M_k^{\text{vis}})$ denotes the region that we would like to inpaint in I_k^e and excludes the region that has been reprojected from the reference view.

Hybrid inpainting and refinement We find that this approach by itself does not work well in our case because the NeRF geometry has errors that lead to reprojection artifacts. Moreover, propagating pixels across large viewpoint differences (especially at oblique views) leads to poor results, notably due to texture stretching. This can be seen in Fig. 3 ($N = 20$).

Instead, we find that it is better to use the reprojected pixels as an *initialization* to the diffusion-based editing process. We achieve this with a novel

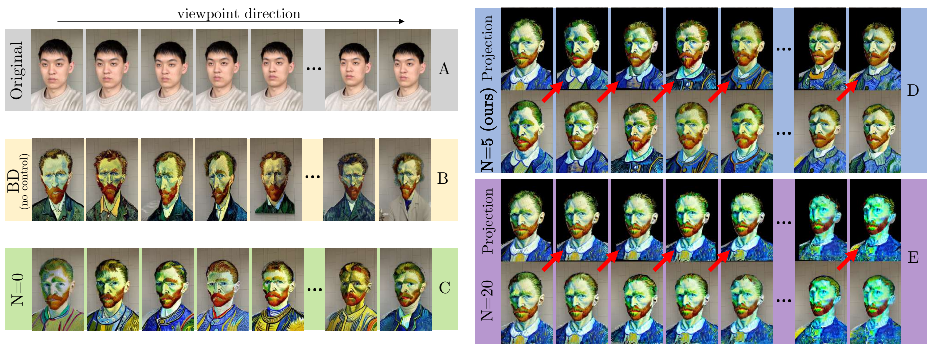

Fig. 3: Projection Inpainting. We analyze our proposed scheme using various views of the input sequence (row A) for the text prompt *"Vincent Van Gogh"*. Frames edited using blended diffusion (row B, BD), without any form of control, align with the prompt but lack both geometric and photometric consistency. Using a depth-aware inpainting model (row C, $N = 0$) achieves geometric alignment but suffers from photometric inconsistency. Iteratively projecting edited images to the next view and only inpainting occluded regions (row E, $N = 20$) produces results that diverge as we get farther from the reference view; we show the projected pixels on top and the inpainted result below. Our hybrid scheme (row D, $N = 5$) balances these two options by starting with the projection result but further refining it to preserve visual quality. Note, minimal inconsistencies are efficiently resolved with NeRF optimization, ensuring improved results.

hybrid inpainting scheme, where we preserve the reprojected pixels for the first $N = 5$ initial denoising steps (Eqn. 3) and fall back to inpainting the entire object regions in subsequent denoising steps (Eqn. 1). These initial diffusion steps thus constrain the overall appearance of the edit and subsequent steps allow the diffusion process to fix disoccluded regions while preserving the flexibility to fix reprojection artifacts. This change of inpainting masks during the diffusion process is illustrated in Fig. 2. See supplementary for pseudocode.

Analysis. We demonstrate the advantages of our approach in Fig. 3 where we show the effect of applying the same text-based edit on a set of input frames. Here N denotes the number of denoising steps (out of a total of 20) that we apply our projection scheme in. $N = 0$ corresponds to no projection at all, i.e., the pure ControlNet-based inpainting scheme described in Sec. 3.2; while the features are coarsely aligned geometrically here, the appearance varies widely from frame to frame. At the other end of the spectrum, $N = 20$ corresponds to retaining projected pixels completely from the reference (first) frame to subsequent frames. While the initial set of edited frames look reasonable, this solution produces poor results as we get further away from the initial viewpoint due to the accumuluation of NeRF geometry errors and resampling issues. Our hybrid approach ($N = 5$ projection inpainting steps followed by refinement of the full masked region) balances these out; it retains higher visual quality at every viewpoint compared to $N = 20$ and has much better consistency than $N = 0$. The remaining minor inconsistencies are easy to fuse in NeRF optimization.

Choice of viewpoints. Our projection inpainting starts by editing a reference viewpoint. This can be user-selected (for example to experiment with the prompt/edit parameters in the best way) or any frame in the input sequence. For every subsequent choice of frame to edit, we use a simple heuristic to maximize overlap between subsequent frames. We re-project pixels within the mask of the current image into the remaining views. The view with the highest number of back-projected pixels is deemed closest. This is repeated for each subsequent image, excluding those already considered, culminating in a sequence of view IDs indicating proximity. The projection inpainting is performed using this sequence.

3.4 Edited NeRF optimization

Once all images have been edited using projection inpainting, we optimize the NeRF (starting from the original NeRF) for 1,000 iterations. This stage transfer the edits in image space into the NeRF. Note that this scheme is in contrast to the "Iterative Dataset Update" approach of Instruct-NeRF2NeRF where each frame is individually edited, followed by 10 iterations of NeRF training. This is required in their approach because individual edits are inconsistent and need to be introduced slowly for NeRF training to converge properly. On the contrary, by ensuring that the individual edits are largely consistent, we are able to edit all images in one go and train the NeRF for a large number of iterations.

After 1000 iterations, the majority of significant NeRF alterations are already accomplished and the images are highly view-consistent. Our focus after this stage is to refine the visual quality further. Hence, we shift to updating the NeRF using the Iterative Dataset Update approach. However, we diverge from their methodology by employing a noise strength between 0.5 and 0.8, in contrast to their choice from 0.02 to 0.98. This generates images that closely resemble the existing ones in terms of major features but are enhanced with finer details. We show that our method leads to much faster convergence than Instruct-NeRF2NeRF in Fig. 6.

3.5 Implementation Details

Our method is implemented within the nerfstudio [45] codebase, utilizing their "nerfacto" model as the underlying NeRF representation. All experiments are conducted using the default hyperparameters: a guidance scale of 7.5 and ControlNet conditioning scale of 0.5. Instruct-NeRF2NeRF uses images of resolution 512×512; we find that ControlNet performs poorly with images at this size. Therefore, to maintain consistency with Instruct-NeRF2NeRF in our experiments, we use this resolution for NeRF training images but bilinearly upsample to double the original dimensions before generation and downsize after.

We run each experiment for 4,000 iterations. As noted before, we run a full round of projection inpainting first, then optimize the input NeRF using the edited images for 1000 iterations. Subsequently, we update individual images independently and intersperse this with 30 iterations of NeRF optimization. For scenes exceeding 150 frames, we edit a maximum of 100 frames. We optimize

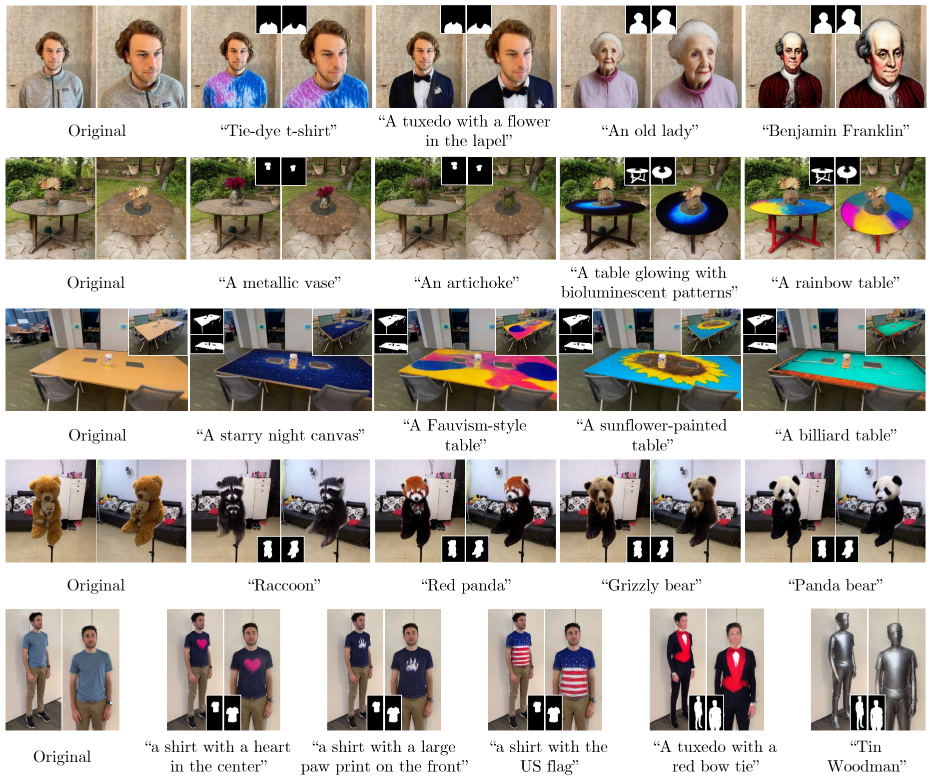

Fig. 4: Results. We present the results of our method on a diverse set of scenes. For each scene, we show input views on the left and results obtained from different text prompts after that.

NeRF with L1 and LPIPS [58] losses. On average, each experiment takes approximately 20 minutes on an NVIDIA A100 GPU.

4 Results

We demonstrate DATENeRF on scenes from Instruct-NeRF2NeRF [13], the *garden* scene from Mip-NeRF 360 [3] and two scenes that we captured ourselves. These scenes vary from largely front-facing captures of people to 360 captures of objects with background (both small-scale and large-scale). We extract masks for user-specified regions of these scenes using the method detailed in Sec. 3.2.

In Figs. 1 and 4, we demonstrate editing results for a subset of these scenes with a variety of text prompts. From the results we can see that our method is able to generate realistic appearance that closely matches the input prompt with high-frequency texture details and consistent geometry. This can be seen from editing the *bear* scene in Fig. 1 to a variety of different animals (note the *"zebra"* edit resulting in clear stripes) as well as retexturing the clothes and surfaces in the scenes in Fig. 4 (note the *"tie-dye t-shirt"* resulting in a clear tie-dye texture and the *"sunflower-painted table"* retaining a clear sunflower design).

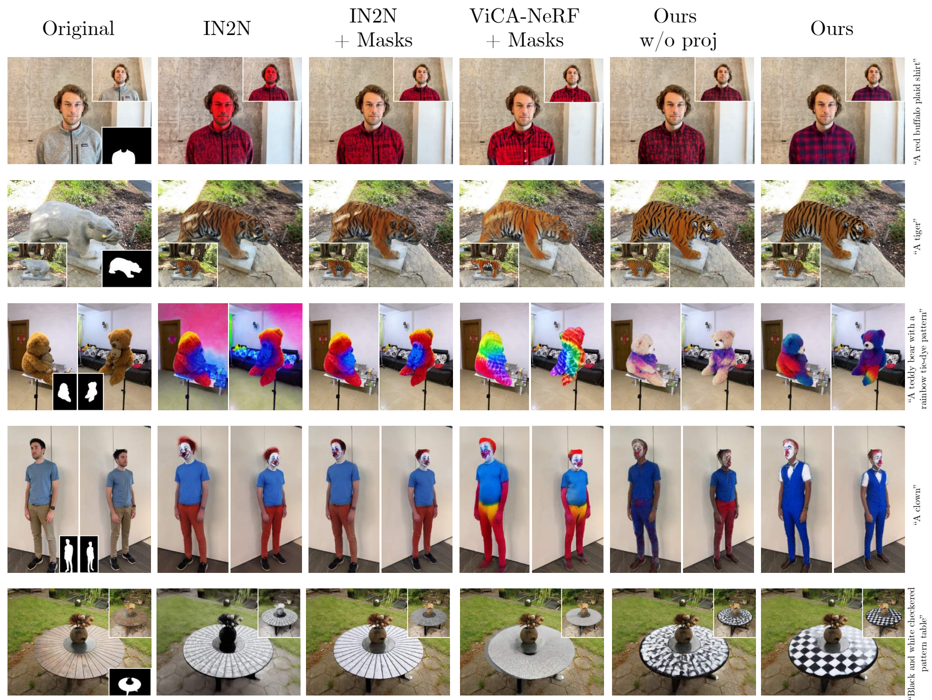

Fig. 5: Comparisons. We compare Instruct-NeRF2NeRF [13], with and without our masks (columns 2 and 3), ViCA-NeRF [11] with our masks (column 4) and our approach both with and without projection inpainting (columns 5 and 6). Our full method allows for drastic and more consistent edits, for e.g., the textures of the plaid shirt and clown costume, the rainbow on the teddy bear, and the checkerboard pattern on the table.

Comparisons with Instruct-NeRF2NeRF and ViCA-NeRF. We compare our method with the state-of-the-art text prompt-based editing methods, Instruct-NeRF2NeRF (IN2N) [13] and ViCA-NeRF [11] in Fig. 5. We generate their results using their code with the default diffusion parameters[3]. For Instruct-NeRF2NeRF, we demonstrate two variations: editing the whole scene (as in their work) and editing only the masked region we use. For ViCA-NeRF, we only present results with our masks, as results without masks have a similar impact to IN2N without them.

To aid NeRF convergence, IN2N makes a number of design choices including conditioning the editing on the input image, adding random amounts of noise and slowly introducing edited images into the NeRF optimization. This tends to retain the appearance of the input images (e.g., *"a red buffalo plaid shirt"* and *"A teddy bear with a rainbow tie-dye pattern"* results) while also being unable to handle high-frequency textures (e.g., *"A tiger"* and *"Black and white checkered pattern table"*). In contrast, by conditioning only on depth and using projection

[3] We use default diffusion parameters for Instruct-NeRF2NeRF, diverging from the original paper where the weights of classifier-free guidance were manually tuned.

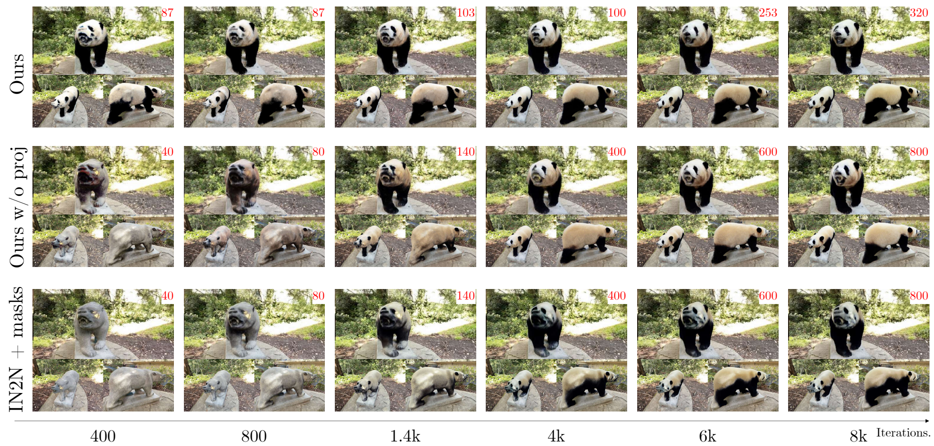

Fig. 6: Convergence Speed. Our method requires fewer iteration and image generation steps to converge compared to Instruct-NeRF2NeRF [13] and ours without projection approach. For both methods, we note the number of diffusion-based image edits being performed (in red in the top right corner) over the course of the NeRF iterations (x-axis at bottom).

inpainting, our method is able to both make drastic edits to the input scene while significantly improving on visual quality and texture detail.

Our method also outperforms ViCA-NeRF that is unable to handle high-frequency textures and produces results that are blurrier (e.g., *"Black and white checkered pattern table"*).

In Fig. 6, we compare the convergence of our method against IN2N on the *bear* scene. DATENeRF edits *all* the 87 images in the scene upfront. As a result, by iteration 400 all images have already been transformed and moreover, as a result of being fairly consistent, result in a clearly edited NeRF model. On the other hand, IN2N has performed 40 image edits, but because many are only slightly edited and moreover are inconsistent, the NeRF at this point is still close to the input scene. In fact, IN2N requires 300 image edits and 3000 NeRF iterations to get results that are qualitatively similar to our results at 87 edits and 400 iterations. Subsequent iterations finetune the quality of our result to capture the detailed, fluffy *"panda bear"* appearance that IN2N is not able to achieve even at 8k iterations.

Ablations. We ablate our projection inpainting scheme by comparing it against an "Ours w/o projection" method. As noted in Sec. 3.2, this method edits individual frames using Blended Diffusion and depth-conditioned ControlNet. As illustrated in Fig. 3, these edits are geometrically reasonably aligned but can vary a lot in appearance. Applied as is, these edited inputs do not allow for the NeRF model to converge. Hence, for this experiment we apply some ideas from IN2N. Specifically, we inject a random amount of noise per image edit using $[\mathrm{t_{min}}, \mathrm{t_{max}}] = [0.8, 0.98]$ for ControlNet (as against $[\mathrm{t_{min}}, \mathrm{t_{max}}] = [0.02, 0.98]$ in IN2N). Also, we edit one image for every 10 NeRF iterations similar to IN2N.

Table 1: Quantitative Evaluation. We evaluate Instruct-NeRF2NeRF, ViCA-NeRF, and our method using different 2D editing models, both with and without projection inpainting. Our full method with ControlNet outperforms both Instruct-NeRF2NeRF variants as well as ViCA-NeRF indicating superior accuracy and uniformity in image rendering from textual prompts and across varied viewpoints.

Method	Image Editing Model	Projection Inpainting	CLIP Text-Image Direction Similarity ↑	CLIP Direction Consistency ↑
Instruct-NeRF2NeRF [13]	Instruct-Pix2Pix [5]		0.1407	0.6349
	ControlNet [57]		0.1330	0.6799
ViCA-NeRF [11]	Instruct-Pix2Pix [5]		0.1683	0.6981
Ours	Instruct-Pix2Pix [5]	✓	0.1618	0.6910
	ControlNet [57]		0.1772	0.6879
	ControlNet [57]	✓	**0.1866**	**0.7069**

This comparison is illustrated in Fig. 5. Here we see that even just using our ControlNet-based scheme already has advantages over IN2N. It performs more drastic (and better text-aligned) changes to the NeRF scene (e.g., the *"Black and white checkered pattern table"* result) and has better quality textures (e.g., the *"Superman clothes"* result) than IN2N. However, the lack of consistency in edits shows up in the final results. Our full method, including the projection inpainting, significantly improves over this, creating crisp geometry and appearance.

Quantitative metrics. We benchmark variants of our method vs. IN2N and ViCA-NeRF in terms of CLIP Text-Image Direction Similarity score and CLIP Direction Consistency for 24 edits in Table 1. The former measures the alignment between the text prompts and the generated images, while the latter assesses the method's ability to maintain consistency when rendering images from different viewpoints. We compare against IN2N and ViCA-NeRF using masks for fairness. The major differences between these approaches and our method are in the base editing model (Instruct-Pix2Pix vs. ControlNet) and the use of projection inpainting in our method. We rigorously evaluate all these variations: the original IN2N with Instruct-Pix2Pix as the image editing model, IN2N with ControlNet instead of Instruct-Pix2Pix, the original ViCA-NeRF, our method with Instruct-Pix2Pix and projection inpainting, our method with ControlNet and no projection and our full method with ControlNet and projection inpainting. Naively adding ControlNet to IN2N worsens CLIP Text-Image Direction Similarity but improves CLIP Direction Consistency. Meanwhile, our method uses InstructPix2Pix as the image editing model in combination with projection inpainting to outperform both versions of IN2N. This indicates that our projection inpainting method can improve the performance even with other image models. Using ControlNet in our method without projection inpainting results in better CLIP Text-Image Direction Similarity but worse CLIP Direction Consistency, indicating poorer view consistency. This is not surprising since the projection inpainting is explicitly designed to make edits view consistent. Our full method outperforms all the other variations including ViCA-NeRF on both metrics producing both better text-aligned edits and better temporal consistency.

Original "A Fauvism painting" "Vincent Van Gogh" "An Edvard Munch Painting"

Fig. 7: Edge-conditioned DATENeRF. We demonstrate that DATENeRF can use controls other than depth such as Canny edges. Edge maps (shown in insets) play a role in maintaining geometric consistency across the rendered scenes. By incorporating the nuanced details captured in the edge maps, DATENeRF is able to interpret the object outlines and structural features into the 3D scene.

Extending to other ControlNet modalities. We demonstrate the flexibility of DATENeRF by experimenting with a different control modality. In Fig. 7, instead of using depth, we use Canny edges as they also carry important geometric information and help preserve details. As we can see, with Canny edge conditioning, the method still produces highly consistent results that preserve the subject pose while aligning very well with the text prompt.

Object Insertion. DATENeRF can also be used for 3D object insertion. Our approach begins with the extraction of the scene's geometry using the technique of TSDF (Truncated Signed Distance Function). With this intermediary geometry established, we can introduce new objects into the scene, as demonstrated in Fig. 8, where we have added a 3D hat model to the person in the scene. We render the depth of the person wearing the hat (shown in the Fig. 8) inset, and a mask for the hat accounting for potential occlusions within the scene by using the NeRF depth. Given these depths and masks as additional inputs, we can use DATENeRF to "render" the hat into the NeRF scene to generated realistic results that maintain the spatial and lighting consistency of the original NeRF. In columns 3 and 4, we first use our method to adapt the original scene to resemble *"Mark Twain"* and *"Albert Einstein"*, then composite the hat to obtain the final results. This form of creative control is only possible with our method because of the use of depth-conditioned inpainting.

Scene Editing. In Fig. 9 we use DATENeRF to edit the entire *garden* scene to produce a painterly rendering in the style of *"Vincent Van Gogh"*.

Limitations. Since our method uses NeRF geometry to make edits consistent, we cannot make large geometric changes to the scene. We also rely on the editing model's capacity to generate content based on the depth maps. For large-scale, complex scenes we find that ControlNet may not always faithfully preserve content that is aligned with the depth map, particularly in the periphery. This is demonstrated in the middle column of Fig. 9. Even so, DATENeRF is able to merge these edits into a consistently edited video, albeit one where the content might not follow the input exactly. This can be potentially addressed using other control signals like edge guidance. Also, we don't model view dependent effects.

Fig. 8: 3D object compositing. We can use DATENeRF to composite 3D objects (a cowboy hat here) into an original or edited NeRFs. The target object is positioned using an intermediary mesh, which is rendered to obtain disparity maps (inset). In columns 3 and 4, we first use our method to adapt the original scene to resemble *"Mark Twain"* and *"Albert Einstein"*, then composite the hat to obtain the final results.

5 Conclusions

In this paper, we introduce DATENeRF, a method to achieve multiview-consistent text-based editing of NeRF scenes. Given a selected object and a text prompt, we achieve complex edits such as material, texture, or content modifications. We leverage a depth-conditioned ControlNet for inpainting and a reprojection scheme using the NeRF scene geometry. We demonstrate realistic, highly detailed, state-of-the-art results on a diverse set of scenes including humans, animals, objects, 360 and front-facing scenes. When compared with existing methods, DATENeRF produces edits that more closely match the text prompts, requires fewer inferences from the diffusion model, and converges more quickly. Moreover, the method's flexibility allows for the use of different types of guidance, such as canny edges or intermediary meshes, broadening its applications. While our method offers many creative possibilities, it also poses ethical concerns. Realistic edits, especially of human faces, can be misused to create misleading or malicious content, raising issues of authenticity and misinformation.

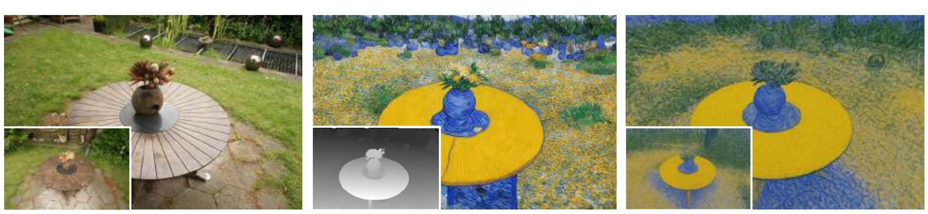

Fig. 9: Scene Editing using DATENeRF. We show two input views (left), the ControlNet edit for one view (with depth in inset) and the final edited result.

Acknowledgements

We thank Duygu Ceylan for advice during the project. We thank anonymous ECCV reviewer 2 for their support and feedback on the paper. The research reported in this publication was partially supported by funding from KAUST Center of Excellence on GenAI, under award number 5940.

References

1. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18208–18218 (2022)
2. Bao, C., Zhang, Y., Yang, B., Fan, T., Yang, Z., Bao, H., Zhang, G., Cui, Z.: Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20919–20929 (2023)
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022)
4. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition (2020)
5. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: CVPR (2023)
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
7. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision (ECCV) (2022)
8. Chen, D.Z., Siddiqui, Y., Lee, H.Y., Tulyakov, S., Nießner, M.: Text2tex: Text-driven texture synthesis via diffusion models. In: ICCV (2023)
9. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. arXiv preprint arXiv:2303.13873 (2023)
10. Chiang, P.Z., Tsai, M.S., Tseng, H.Y., Lai, W.S., Chiu, W.C.: Stylizing 3d scene via implicit representation and hypernetwork. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1475–1484 (2022)
11. Dong, J., Wang, Y.X.: Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. Advances in Neural Information Processing Systems **36** (2024)
12. Gordon, O., Avrahami, O., Lischinski, D.: Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. arXiv preprint arXiv:2306.12760 (2023)
13. Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. arXiv preprint arXiv:2303.12789 (2023)
14. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE transactions on pattern analysis and machine intelligence **35**(6), 1397–1409 (2012)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. arXiv preprint arxiv:2006.11239 (2020)
16. Huang, H.P., Tseng, H.Y., Saini, S., Singh, M., Yang, M.H.: Learning to stylize novel views. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13869–13878 (2021)

17. Huang, Y.H., He, Y., Yuan, Y.J., Lai, Y.K., Gao, L.: Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18342–18352 (2022)
18. Jaganathan, V., Huang, H.H., Irshad, M.Z., Jampani, V., Raj, A., Kira, Z.: Ice-g: Image conditional editing of 3d gaussian splats (2024)
19. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields (2022)
20. Jambon, C., Kerbl, B., Kopanas, G., Diolatzis, S., Leimkühler, T., Drettakis, G.: Nerfshop: Interactive editing of neural radiance fields". Proceedings of the ACM on Computer Graphics and Interactive Techniques **6**(1) (May 2023), `https://repo-sam.inria.fr/fungraph/nerfshop/`
21. Kerr, J., Kim, C.M., Goldberg, K., Kanazawa, A., Tancik, M.: Lerf: Language embedded radiance fields. In: International Conference on Computer Vision (ICCV) (2023)
22. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv:2304.02643 (2023)
23. Kobayashi, S., Matsumoto, E., Sitzmann, V.: Decomposing nerf for editing via feature field distillation. Advances in Neural Information Processing Systems **35**, 23311–23330 (2022)
24. Kuang, Z., Luan, F., Bi, S., Shu, Z., Wetzstein, G., Sunkavalli, K.: Palettenerf: Palette-based appearance editing of neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20691–20700 (2023)
25. Kuang, Z., Olszewski, K., Chai, M., Huang, Z., Achlioptas, P., Tulyakov, S.: Neroic: Neural rendering of objects from online image collections. ACM Trans. Graph. **41**(4) (jul 2022)
26. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
27. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023)
28. Liu, S., Zhang, X., Zhang, Z., Zhang, R., Zhu, J.Y., Russell, B.: Editing conditional radiance fields. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021)
29. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: Inpainting using denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
30. Mikaeili, A., Perel, O., Safaee, M., Cohen-Or, D., Mahdavi-Amiri, A.: Sked: Sketch-guided text-based 3d editing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14607–14619 (2023)
31. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
32. Mirzaei, A., Aumentado-Armstrong, T., Derpanis, K.G., Kelly, J., Brubaker, M.A., Gilitschenski, I., Levinshtein, A.: SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In: CVPR (2023)

33. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (Jul 2022)
34. Nguyen-Phuoc, T., Liu, F., Xiao, L.: Snerf: stylized neural implicit representations for 3d scenes. arXiv preprint arXiv:2207.02363 (2022)
35. Peng, Y., Yan, Y., Liu, S., Cheng, Y., Guan, S., Pan, B., Zhai, G., Yang, X.: Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation. Advances in Neural Information Processing Systems **35**, 31402–31415 (2022)
36. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: ICLR (2023)
37. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision (2021)
38. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents (2022)
39. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE transactions on pattern analysis and machine intelligence **44**(3), 1623–1637 (2020)
40. Richardson, E., Metzer, G., Alaluf, Y., Giryes, R., Cohen-Or, D.: Texture: Text-guided texturing of 3d shapes (2023)
41. Rojas, S., Zarzar, J., Pérez, J.C., Sanakoyeu, A., Thabet, A., Pumarola, A., Ghanem, B.: Re-rend: Real-time rendering of nerfs across devices. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3632–3641 (2023)
42. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
43. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Gontijo-Lopes, R., Ayan, B.K., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems (2022)
44. Sella, E., Fiebelman, G., Hedman, P., Averbuch-Elor, H.: Vox-e: Text-guided voxel editing of 3d objects. In: Proceedings of the International Conference on Computer Vision (ICCV) (2023)
45. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
46. Wang, C., Chai, M., He, M., Chen, D., Liao, J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3835–3844 (2022)
47. Wang, C., Jiang, R., Chai, M., He, M., Chen, D., Liao, J.: Nerf-art: Text-driven neural radiance fields stylization. IEEE Transactions on Visualization and Computer Graphics (2023)
48. Wang, D., Zhang, T., Abboud, A., Süsstrunk, S.: Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. arXiv preprint arXiv:2305.15094 (2023)
49. Wang, H., Du, X., Li, J., Yeh, R.A., Shakhnarovich, G.: Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In: CVPR (2023)

50. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)
51. Wu, Q., Liu, X., Chen, Y., Li, K., Zheng, C., Cai, J., Zheng, J.: Object-compositional neural implicit surfaces. In: European Conference on Computer Vision. pp. 197–213. Springer (2022)
52. Wu, Q., Wang, K., Li, K., Zheng, J., Cai, J.: Objectsdf++: Improved object-compositional neural implicit surfaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 21764–21774 (2023)
53. Wu, Q., Tan, J., Xu, K.: Palettenerf: Palette-based color editing for nerfs. arXiv preprint arXiv:2212.12871 (2022)
54. Yu, L., Xiang, W., Han, K.: Edit-diffnerf: Editing 3d neural radiance fields using 2d diffusion model (2023)
55. Yuan, Y.J., Sun, Y.T., Lai, Y.K., Ma, Y., Jia, R., Gao, L.: Nerf-editing: geometry editing of neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18353–18364 (2022)
56. Zhang, K., Kolkin, N., Bi, S., Luan, F., Xu, Z., Shechtman, E., Snavely, N.: Arf: Artistic radiance fields. In: ECCV. pp. 717–733. Springer (2022)
57. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3836–3847 (2023)
58. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
59. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: neural factorization of shape and reflectance under an unknown illumination. ACM Trans. Graph. **40**(6) (dec 2021)
60. Zhuang, J., Wang, C., Liu, L., Lin, L., Li, G.: Dreameditor: Text-driven 3d scene editing with neural fields. arXiv preprint arXiv:2306.13455 (2023)