SHAP-EDITOR: Instruction-guided Latent 3D Editing in Seconds

Minghao Chen Junyu Xie Iro Laina Andrea Vedaldi
Visual Geometry Group, University of Oxford
{minghao, jyx, iro, vedaldi}@robots.ox.ac.uk
silent-chen.github.io/Shap-Editor

Figure 1. Given 3D assets as inputs, SHAP-EDITOR achieves fast 3D editing in just one second by learning a feed-forward mapping in the latent space of a 3D asset generator.

Abstract

We propose a novel feed-forward 3D editing framework called SHAP-EDITOR. *Prior research on editing 3D objects primarily concentrated on editing individual objects by leveraging off-the-shelf 2D image editing networks, utilizing a process called 3D distillation, which transfers knowledge from the 2D network to the 3D asset. Distillation necessitates at least tens of minutes per asset to attain satisfactory editing results, thus it is not very practical. In contrast, we ask whether 3D editing can be carried out directly by a feed-forward network, eschewing test-time optimization. In particular, we hypothesise that this process can be greatly simplified by first encoding 3D objects into a suitable latent space. We validate this hypothesis by building upon the latent space of Shap-E. We demonstrate that direct 3D editing in this space is possible and efficient by learning a feed-forward editor network that only requires approximately one second per edit. Our experiments show that* SHAP-EDITOR *generalises well to both in-distribution and out-of-distribution 3D assets with different prompts and achieves superior performance compared to methods that carry out test-time optimisation for each edited instance.*

1. Introduction

We consider the problem of generating and editing 3D objects based on instructions expressed in natural language. With the advent of denoising diffusion models [20, 56, 62, 64], text-based image generation [56] and editing [3, 18, 49] have witnessed remarkable progress. Many authors have since attempted to transfer such capabilities to 3D via *test-time optimisation*, where a 3D model is optimised from scratch until its rendered 2D appearance satisfies an underlying prior captured by a pre-trained 2D models [17, 24, 59].

While optimisation-based methods obtain encouraging results, they are not scalable — in fact, a single 3D generation or edit can take from minutes to hours. It is thus natural to seek more efficient generators and editors that can directly work in 3D. We hypothesise that this can be greatly facilitated by first learning a suitable *latent space* for 3D models. Such an approach is exemplified by Shape-E [23], a conditional generative model that learns an auto-encoder mapping 3D objects into vectors (latents). For generation, these vectors can be sampled *directly* by a diffusion model, eschewing test-time optimisation entirely.

In this paper, we explore the capability of a 3D latent space to not only facilitate 3D generation but also enable efficient 3D editing. To this end, we propose SHAP-EDITOR, a method that enables semantic, text-driven edits directly in the latent space of 3D asset generators. Because of the properties of the latent space, once learned, the editor function is capable of applying the edit to any new object in just one second, in contrast to the several minutes or even hours required by optimisation-based approaches.

In more detail, our method starts from a *3D auto-encoder* — *e.g.*, the off-the-shelf Shape-E encoder. It also takes as input a *2D image editor* that can understand instructions in natural language. For any such instruction, SHAP-EDITOR learns a function that can map, in a feed-forward manner, the latent of any 3D object into the latent of the corresponding edit — we call this a *latent editor*. The fact that the latent editor can be learned relatively easily is a strong indication that the 3D latent space has a useful structure for this type of operations. Empirically, we further explore and demonstrate the partial linearity of such edits when they are carried out in this space.

SHAP-EDITOR has several interesting practical properties. First, we learn a single latent editor that works universally for any input object. This function lifts the knowledge contained in the 2D image editor to the 3D space via distillation losses. In fact, we show that we can distill simultaneously *several* different 2D editors of complementary strengths. In our student-teacher framework, the *combined* knowledge of the editors is then transferred to the latent editor.

Second, we note that the latent editor is able to capture certain semantic concepts, and in particular complex compositions of concepts, better than the original text-to-3D generator. Moreover, it allows the application of several edits sequentially, with cumulative effects.

Third, while our method learns an editor function for each type of edit, at test time it can be applied to any number of objects very quickly, which could be used to modify libraries of thousands of 3D assets (*e.g.*, to apply a style to them). In this sense, it can be seen as an amortised counterpart to methods that use test-time optimisation. We also demonstrate that, by conditioning the latent editor on text, several different edits can be learned successfully by a single model. This suggests that, given sufficient training resources, it might be possible to learn an open-ended editor.

To summarise, our contributions are: (1) We show that 3D latent representations of objects designed for generation can also support semantic editing; (2) We propose a method that can distill the knowledge of one or more 2D image generators/editors in a single latent editor function which can apply an edit in seconds, significantly reducing the computational costs associated with test-time optimisation; (3) We show that this latent function does better at compositional tasks than the original 3D generator; (4) We further show that it is possible to extend the latent editor to understand multiple editing instructions simultaneously.

2. Related Work

Diffusion-based image manipulation. Recent advances in text-guided diffusion models have greatly improved 2D image generation. Yet, these models typically offer limited control over the generated content. To enable controllable generation, researchers have explored concept personalisation [13, 31, 57], layout control [4, 6, 10, 37], and other conditionings [86]. Other recent works [2, 18, 27, 42, 49, 52, 68] have extended text-guided diffusion models to image-to-image translation tasks and image editing. InstructPix2Pix (IP2P) [3] finetunes a diffusion model to accept image conditions and instructional prompts as inputs, by training on a large-scale synthetic dataset. Subsequent research [85, 87] has sought to further finetune InstructPix2Pix with manually annotated datasets.

Neural field manipulation. Several attempts have been made to extend neural fields, such as NeRFs [46], with editing capabilities. EditNeRF [39] was the first approach to edit the shape and color of a NeRF given user scribbles. Approaches that followed include 3D editing from just a single edited view [1], or via 2D sketches [45], keypoints [88], attributes [25], meshes [22, 53, 77, 79, 81] or point clouds [5]. Others focus on object removal with user-provided points or masks [47, 66, 74], object repositioning [78], recoloring [15, 30, 32] and style transfer [76, 84].

Text-to-3D generation. Given the success of diffusion-based generation and vision-language models such as CLIP [55], several methods have been proposed for generating 3D scenes using text prompts [21, 48]. A pioneering work is DreamFusion [54], which proposes the Score Distillation Sampling (SDS) loss. They use it to optimise a parametric model, such as NeRF, with the supervision of an off-the-shelf 2D diffusion model. DreamFusion has since been improved by follow-up work [7, 38, 43, 73], but these methods are generally not directly applicable to 3D editing tasks. ATT3D [40] further reduces the inference time by amortising the distillation time to the training phase. Another direction is to train auto-encoders on explicit 3D representations, such as point clouds [82] or voxel grids [58], or on implicit functions, such as signed distance functions [11] or neural radiance fields [29]. The generative models are typically trained in the latent space [8, 50] and are conditioned on text inputs. The most related work is Shap-E [23] which trains on a very large-scale dataset (several millions). It encodes 3D assets into latents that are directly decoded to output NeRF parameters, signed distance functions, or texture fields [14, 61]. It also incorporates a diffusion model [20] to facilitate conditional 3D asset generation.

Text-based 3D editing. Differently from text-to-3D generation, editing methods start from a given 3D object or scene (usually represented by a NeRF [46] or voxel grid [65]). Some authors leverage CLIP embeddings or similar models [34, 35, 41] to perform text-driven semantic editing/stylisation globally [33, 44, 70, 71] or locally [16, 28, 63, 72]. For instance, CLIP-NeRF [70] learns a class-specific NeRF and latent spaces (*e.g.*, for cars, chairs) and encodes objects in latent space via optimisation.

Most recent and concurrent approaches leverage diffu-

sion priors. Starting with Instruct-NeRF2NeRF [17], one line of research employs pre-trained 2D models to edit image renderings of the original model and uses these to gradually update the underlying 3D representation [17, 69, 80]. Instead of editing images, others optimise the 3D representation directly with different variants of score distillation sampling [9, 24, 36, 51, 59, 83, 89, 90]. They often differ in their use of the diffusion prior; *e.g.*, [17, 24] use InstructPix2Pix [3], while most others rely on Stable Diffusion [20]. Many existing methods edit scenes globally, which may sometimes affect unintended regions. To address this issue, approaches such as Vox-E [59] and Focal-Dreamer [36], introduce mechanisms for local 3D editing. We note, however, that, due to their inherent design, most methods cannot handle global and local edits equally well.

In contrast, we show that we can train a *single* network for *both* types of edits with a loss tailored to each edit type. We also note that all these methods perform editing via test-time optimisation, which does not allow interactive editing in practice; [69, 80] focus on accelerating this process, but they still use an optimisation-based approach. Instead, our feed-forward network applies edits instantaneously.

3. Method

In order to generate or edit a 3D object, we must first choose a suitable representation θ, specifying its shape and appearance. Common choices for θ include textured meshes and radiance fields, but these are often difficult to use directly in semantic tasks such as text-driven 3D generation and editing. For images, analogous tasks are often simplified by adopting a latent representation. In this paper, we thus ask whether replacing θ with a corresponding latent code $\boldsymbol{r}$ can result in similar benefits for 3D editing.

More formally, we consider the problem of constructing an *editor* function $f : (\theta^s, y) \mapsto \theta^e$ which takes as input a 3D object θ^s (source) and produces as output a new version of it θ^e (edit) according to natural-language instructions y. For example, θ^s could be the 3D model of a *corgi*, y could say *"Give it a Christmas hat"*, then θ^e would be the *same corgi* but with the *hat*.

Learning the map f directly is challenging because interpreting natural language in an open-ended manner requires large models trained on billions of data samples, which are generally not available in 3D. Some authors have approached this problem by starting from existing 2D image models, trained on billions of images. We can think of a 2D editor as a conditional distribution $p(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y)$ of possible edits $\boldsymbol{x}^e$ given the source image $\boldsymbol{x}^s$. Then, one can obtain θ^e by optimising the log-posterior $\mathbb{E}_\pi \left[\log p(\mathcal{R}(\theta^e, \pi) \,|\, \boldsymbol{x}^s, y)\right]$ where $\mathcal{R}(\theta^e, \pi)$ is the image obtained by rendering θ^e from a random viewpoint π with a differentiable renderer $\mathcal{R}$. This, however, requires *per-instance optimisation at test time*, so obtaining θ^e may take minutes to hours in practice.

Here, we thus study the problem of learning a much faster *feed-forward editor* function f. To do so, we first consider a pair of encoder-decoder functions $h : \theta \mapsto \boldsymbol{r}$ and $h^* : \boldsymbol{r} \mapsto \theta$, mapping the 3D object θ to a corresponding latent representation $\boldsymbol{r}$. We then learn a *latent editor* $g : (\boldsymbol{r}^s, y) \mapsto \boldsymbol{r}^e$ which performs the *edit directly in latent space*. Hence, we decompose the editor as $f|_y = h^* \circ g|_y \circ h$. This can be advantageous if, by exploiting the compactness and structure of the latent space, the latent editor $g|_y$ can be fast, efficient, and easy to learn.

In the rest of the section, we review important background (Sec. 3.1), explain how we build and train the latent editor (Sec. 3.2), and finally describe a combination of 2D priors for global and local edits (Sec. 3.3).

3.1. Background

Shap-E: an off-the-shelf 3D latent space. Instead of learning a latent space from scratch, we turn to a pre-trained off-the-shelf model, Shap-E [23], which is a conditional generative model of 3D assets that utilises a latent space. It comprises an auto-encoder that maps 3D objects to latent codes as well as a diffusion-based text/image-conditioned generator that operates in said space. In our work, we mainly use the encoder/decoder components, denoted as h and h^*, respectively, mapping the 3D object from/to a latent vector $\boldsymbol{r} \in \mathbb{R}^{1024 \times 1024}$. In an application, the source latent $\boldsymbol{r}^s$ can be either obtained using h starting from a mesh, or can be sampled from a textual description using the Shape-E generator. For more details, please refer to the supp. mat.

Score Distillation Sampling (SDS). SDS [54] is a loss useful for distilling diffusion probabilistic models (DPMs). Recall that a DPM models a data distribution $p(\boldsymbol{x})$ by learning a denoising function $\boldsymbol{\epsilon} \approx \hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_t; y, t)$, where $\boldsymbol{x}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}$ is a noised version of the data sample $\boldsymbol{x}$. Here (α_t, σ_t) define the noise schedule, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ is normally distributed, and $t = 0, 1, \ldots, T$ are noising steps. The SDS energy function is given by $\mathcal{L}_{\text{SDS}}(\boldsymbol{x}) = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[-\sigma_t \log p(\boldsymbol{x}_t)\right]$, where $p(\boldsymbol{x}_t)$ is the noised version of the data distribution $p(\boldsymbol{x})$ and the noise level is sampled according to a distribution $w(t)$. The reason for choosing this distribution is that the denoising function is also an estimator of the gradient of the log-posterior $\log p(\boldsymbol{x}_t; y, t)$, in the sense that $\hat{\boldsymbol{\epsilon}}(\boldsymbol{x}_t; y, t) = -\sigma_t \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t; y, t)$. Hence, one obtains the (variance controlled) gradient estimator

$$\nabla_{\boldsymbol{x}} \mathcal{L}_{\text{SDS}}(\boldsymbol{x}) = \mathbb{E}_{t,\boldsymbol{\epsilon}} \Big[\hat{\boldsymbol{\epsilon}}\left(\boldsymbol{x}_t; y, t\right) - \boldsymbol{\epsilon}\Big] \tag{1}$$

For 3D distillation, $\boldsymbol{x} = \mathcal{R}(\theta, \pi)$, so the chain rule is used to compute the gradient w.r.t. θ and the loss is also averaged w.r.t. random viewpoints π.

3.2. 3D Editing in Latent Space

We now consider the problem of learning the latent editor g (*i.e.*, our SHAP-EDITOR), using the method summarised

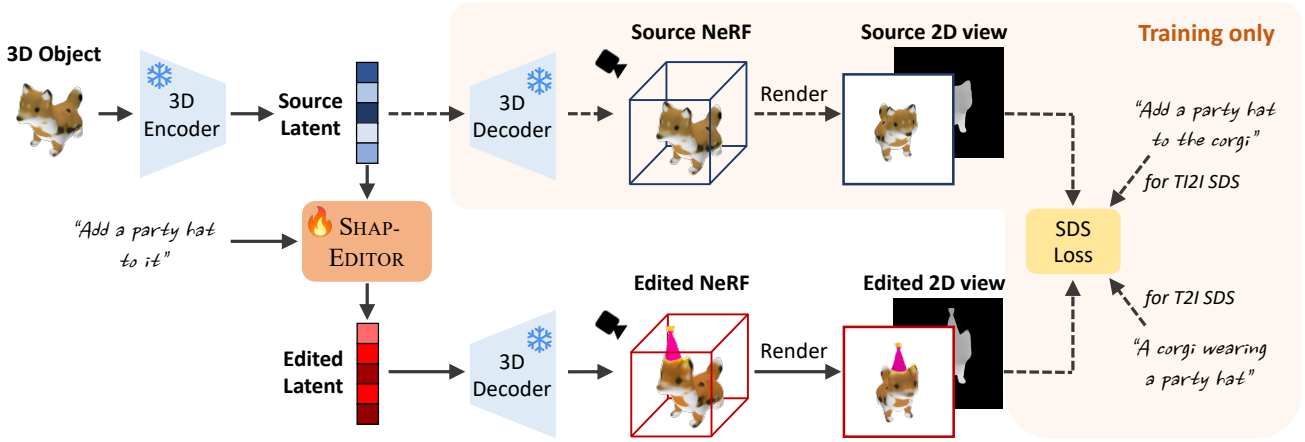

Figure 2. **Latent 3D editing with SHAP-EDITOR.** During training, we use the Shap-E encoder to map a 3D object into a latent space. The source latent and a natural language instruction are then fed into an editing network that produces an edited latent. The edited latent and original latent are decoded into NeRFs and we render a pair of views (RGB images and depth maps) with the same viewpoint for the two different NeRFs. The paired views are used for distilling knowledge from pre-trained 2D generators (T2I) and editors (TI2I) into SHAP-EDITOR. During inference, one only needs to pass the latent code to SHAP-EDITOR, resulting in fast, feed-forward editing.

in Fig. 2 and Algorithm 1. Learning such a function would require suitable triplets (θ^s, θ^e, y) consisting of source and target 3D objects and the instructions y, but there is no such dataset available. Like prior works that use test-time optimisation, we start instead from an existing 2D editor implementing the posterior distribution $p(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y)$, but we only use it for supervising g at training time, not at test time. An additional benefit is that this approach can fuse the knowledge contained in different 2D priors into a single model, which, as we show later, may be better suited for different kinds of edits (*e.g.*, local vs global).

Training the latent editor. Training starts from a dataset Θ of source 3D objects θ^s which are then converted in corresponding latent codes $\boldsymbol{r}^s = h(\theta^s)$ by utilising the encoder function h or sampling the text-to-3D generator $p(\boldsymbol{r}^s \mid y^s)$ given source descriptions y^s.

The latent editor $\boldsymbol{r}^e = g(\boldsymbol{r}^s, y)$ is tasked with mapping the source latent $\boldsymbol{r}^s$ to an edited latent $\boldsymbol{r}^e$ based on the instructions y. We supervise this function with a 2D editor (or mixture of editors) providing the conditional distribution $p(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y)$ using the SDS loss, which in the case can be written as:

$$\mathcal{L}_{\text{SDS-E}}(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[-\sigma_t \log p(\boldsymbol{x}_t^e \mid \boldsymbol{x}^s, y)\right], \quad (2)$$

where $\boldsymbol{x}_t^e = \alpha_t \boldsymbol{x}^e + \sigma_t \boldsymbol{\epsilon}$, and $\boldsymbol{x}^s = \mathcal{R}(h^*(\boldsymbol{r}^s), \pi)$ and $\boldsymbol{x}^e = \mathcal{R}(h^*(\boldsymbol{r}^e), \pi)$ are renders of the object latents $\boldsymbol{r}^s$ and $\boldsymbol{r}^e$, respectively, from a randomly-sampled viewpoint π. Importantly, the rendering functions are differentiable.

We choose this loss because its gradient can be computed directly from any DPM implementation of the 2D editor (Sec. 3.1). At every learning iteration, a new source latent $\boldsymbol{r}^s$ is considered, the edited image $\boldsymbol{x}^e = \mathcal{R}(g(\boldsymbol{r}^s, y), \pi)$ is obtained, and the gradient $\nabla_{\boldsymbol{x}^e} \mathcal{L}_{\text{SDS-E}}(\boldsymbol{x}^e \mid \boldsymbol{x}^s, y)$ is back-propagated to g to update it.

Algorithm 1 SHAP-EDITOR training

Input: Θ: training 3D objects
g: latent editor initialization (h, h^*): auto-encoder,
$\mathcal{L}$: distillation loss $\mathcal{Y}$: instruction set
Output: g: optimized editor
while not converged **do**
  $\boldsymbol{r}^s \leftarrow h(\theta^s),\ \theta^s \in \Theta$
  $\boldsymbol{r}^e \leftarrow g(\boldsymbol{r}^s, y),\ y \in \mathcal{Y}$
  ▷ Render objects to RGB and depth
  $\pi \leftarrow$ random viewpoint
  $(\boldsymbol{x}^s, \boldsymbol{d}^s) \leftarrow \mathcal{R}(h^*(\boldsymbol{r}^s), \pi)$
  $(\boldsymbol{x}^e, \boldsymbol{d}^e) \leftarrow \mathcal{R}(h^*(\boldsymbol{r}^e), \pi)$
  Update g using the gradient $\Delta_g \mathcal{L}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e)$
end while

In practice, we utilise a loss that combines gradients from one or more 2D image editors, thus combining their strengths. Likewise, we can incorporate in this loss additional regularisations to improve the quality of the solution. Here we consider regularising the depth of the edited shape and appearance of the rendered image. We discuss these in detail in Sec. 3.3.

The choice of g. Rather than learning the editing function g from scratch, we initialise g from Shap-E using the same architecture and pre-trained weights. In particular, we note that Shape-E provides a transformer-based denoising neural network that maps a noised code $\boldsymbol{r}_\tau^s = \alpha_\tau \boldsymbol{r}^s + \sigma_\tau \boldsymbol{\epsilon}$ to an estimate $\boldsymbol{r}^s \approx \hat{\boldsymbol{r}}_{\text{SE}}(\boldsymbol{r}_\tau^s; y, \tau)$ of the original latent. We thus set $g(\boldsymbol{r}^e \mid \boldsymbol{r}^s, y) = \hat{\boldsymbol{r}}_{\text{SE}}(\boldsymbol{r}, \tau, y)$, as an initialisation, where $\boldsymbol{r} = \texttt{Concat}(\alpha_\tau \boldsymbol{r}^s + \sigma_\tau \boldsymbol{\epsilon},\ \boldsymbol{r}^s)$ is obtained by stacking the noised input $\boldsymbol{r}_\tau^s$ for a fixed noise level with the original latent $\boldsymbol{r}^s$, since Shap-E originally expects a noisy input. We note that the learned distribution in the original Shap-E is very different from the desired editing distribution.

3.3. 2D Editors

We consider two types of edits: (i) **global edits** (*e.g.*, *"Make it look like a statue"*), which change the style of the object but preserve its overall structure, and (ii) **local edits** (*e.g.*, *"Add a party hat to it"*), which change the structure of the object locally, but preserve the rest. To achieve these, we learn our latent editor from a combination of complementary 2D editors and regularisation losses. For both edit types, we adopt a text-guided image-to-image (TI2I) editor for distillation and consider further edit-specific priors.

3.3.1 Global Editing

TI2I loss. To distill a pre-trained TI2I editor (*e.g.*, InstructPix2Pix [3]) into g, we obtain the SDS gradient $\nabla_{\boldsymbol{x}_e}\mathcal{L}_{\text{SDS-TI2I}}(\boldsymbol{x}^e \mid \boldsymbol{x}^s, y)$, where $\mathcal{L}_{\text{SDS-TI2I}}$ follows Eq. (2), from the TI2I denoising network $\hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}_t^e; \boldsymbol{x}^s, y, t)$. Note that $\hat{\boldsymbol{\epsilon}}_{\text{TI2I}}$ is conditioned on the source image $\boldsymbol{x}^s$ and the editing instruction y. We also use classifier-free guidance (CFG) [19] to enhance the signal of this network for distillation purposes. Please refer to the supp. mat. for details.

Depth regularisation for global editing. Global edits are expected to change the style or appearance of an object, but to retain its overall shape. We further encourage this behaviour via an additional depth regularisation loss:

$$\mathcal{L}_{\text{reg-global}}(\boldsymbol{d}^e, \boldsymbol{d}^s) = \mathbb{E}_\pi\left[\|\boldsymbol{d}^e - \boldsymbol{d}^s\|^2\right], \tag{3}$$

where $\boldsymbol{d}^e$ and $\boldsymbol{d}^s$ are the rendered depth maps from a viewpoint π for the edited and source objects, respectively.

Overall loss. We train SHAP-EDITOR for global edits using $\mathcal{L}_{\text{global}}, (\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e)$, a weighted combination of $\mathcal{L}_{\text{SDS-TI2I}}$ and $\mathcal{L}_{\text{reg-global}}$.

3.3.2 Local Editing

For local edits, we use $\mathcal{L}_{\text{SDS-TI2I}}$ as before, but also consider additional inductive priors, as follows.

T2I loss. Current 2D editors such as InstructPix2Pix often struggle to edit images *locally*, sometimes failing to apply the edit altogether. To encourage semantic adherence to the edit instruction, we further exploit the semantic priors in text-to-image (T2I) models, obtaining the SDS gradient $\nabla_{\boldsymbol{x}_e}\mathcal{L}_{\text{T2I}}(\boldsymbol{x}^e \mid y^e)$ from the denoising network $\hat{\boldsymbol{\epsilon}}_{\text{T2I}}(\boldsymbol{x}_t^e; y^e, t)$. Here, the text prompt y^e contains a full description of the edited object (*e.g.*, *"A corgi wearing a party hat"*), instead of an instruction based on a reference image. We use CFG for this gradient as well.

Masked regularisation for local editing. To further enhance the locality of the edits, we adopt a local regularisation loss inspired by the cross-attention guidance proposed for controllable generation [6, 10]. Specifically, we extract the cross-attention maps from the pre-trained TI2I model during the SDS loss calculation. For instance, given a local editing instruction *"Add a party hat to the corgi"*, we compute the cross-attention maps between the image features and the specific text embedding for the word *"hat"*. These maps are then processed to yield a mask $\boldsymbol{m}$, which represents an estimation of the editing region.

We can then use the complement of the mask to encourage the appearance of the source and edited objects to stay constant outside of the edited region:

$$\mathcal{L}_{\text{reg-local}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e, \boldsymbol{m}) = \mathbb{E}_\pi\Big[(1-\boldsymbol{m}) \odot \left(\lambda_{\text{photo}}\|\boldsymbol{x}^e - \boldsymbol{x}^s\|^2 + \lambda_{\text{depth}}\|\boldsymbol{d}^e - \boldsymbol{d}^s\|^2\right)\Big], \tag{4}$$

where λ_{photo} and λ_{depth} denote corresponding weight factors for the photometric loss $\|\boldsymbol{x}^e - \boldsymbol{x}^s\|^2$ and the depth map differences $\|\boldsymbol{d}^e - \boldsymbol{d}^s\|^2$ between source and edited views.

Overall loss. We train SHAP-EDITOR for local edits using $\mathcal{L}_{\text{local}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e, \boldsymbol{m})$, a weighted combination of $\mathcal{L}_{\text{SDS-TI2I}}$, $\mathcal{L}_{\text{SDS-T2I}}$, and $\mathcal{L}_{\text{reg-local}}$.

4. Experiments

In this section, we provide details of our implementation and the evaluation dataset, compare different variants of our approach to state-of-the-art instruction-based 3D editing methods, and study the effect of the various losses in our approach via ablation.

4.1. Dataset and implementation details

Dataset. We construct our 3D object dataset from two sources: (i) scanned 3D objects from OmniObject3D [75], and (ii) 3D objects generated by Shap-E for specific object categories. To ensure the high quality of synthetic 3D objects, we apply additional filtering based on their CLIP scores. The resultant training dataset encompasses approximately 30 classes, each containing up to 10 instances. For evaluation, we set up 20 instance-instruction pairs. These pairs are composed of 5 editing instructions (3 for global editing and 2 for local editing) and 15 high-quality 3D objects *which are not included in the training set*.

Evaluation metrics. Following common practice [36, 59], we assess edits by measuring the alignment between the generated results and the editing instructions using CLIP similarity (CLIP_{sim}) and CLIP directional similarity [12] (CLIP_{dir}). CLIP_{sim} is the cosine similarity between the edited output and the target text prompts. CLIP_{dir} first calculates the editing directions (*i.e.*, {target vectors minus source vectors}) for both rendered images and text descriptions, followed by the evaluation of the cosine similarity between these two directions. Additionally, to assess structural consistency in *global editing*, we utilise the Structure Distance proposed by [67]. This is the cosine similarity between the self-attention maps generated by two images.

Model	Per-instance optimization	Local editing		Global editing			Inference time ↓
		CLIP$_{sim}$ ↑	CLIP$_{dir}$ ↑	CLIP$_{sim}$ ↑	CLIP$_{dir}$ ↑	Structure Dist. ↓	
Text2Mesh [44]	✓	0.239	0.058	0.248	0.057	0.073	∼ 14 min
Instruct-NeRF2NeRF [17]	✓	0.253	0.051	0.239	0.057	0.095	∼ 36 min
Vox-E [59]	✓	0.277	0.075	0.271	0.066	0.026	∼ 40 min (+ 13 min)
Ours (Test-time Opt.)	✓	0.290	0.087	0.268	**0.072**	0.013	∼ 19 min
Ours (Single-prompt)	✗	**0.292**	**0.097**	**0.272**	0.070	**0.008**	∼ **1 sec**
Ours (Multi-prompt)	✗	0.279	0.085	0.255	0.062	0.009	∼ **1 sec**

Table 1. **Quantitative comparison of our SHAP-EDITOR with other per-instance editing methods**. The measured inference time excludes both the rendering process and the encoding of 3D representations. The time inside the bracket indicates the extra time required by Vox-E for its refinement step in local editing. Our method achieves superior results within one second on the evaluation dataset.

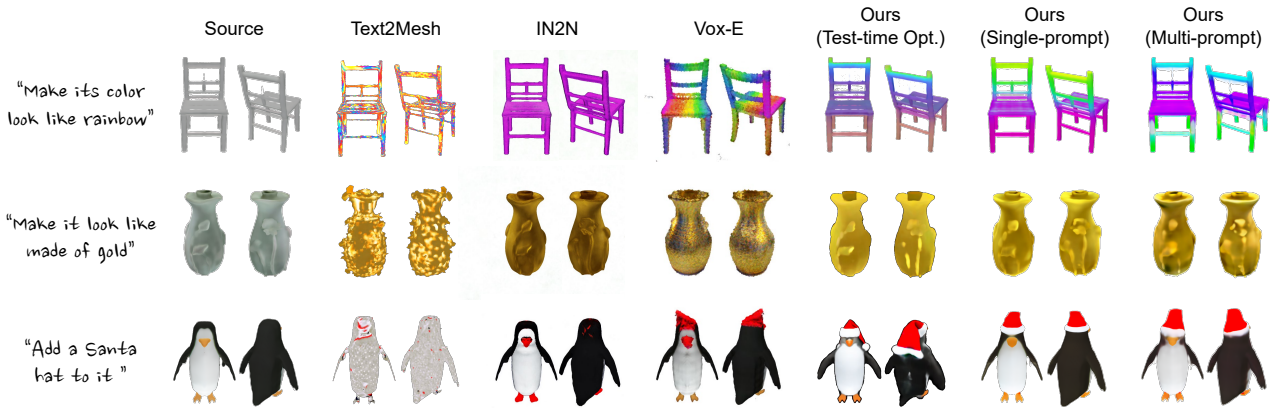

Figure 3. **Qualitative comparison with text-guided 3D editing methods.** Both the single-prompt and multi-prompt versions of our method achieve superior local and global editing results. Our method can preserve the identity of the original assets, such as the appearance and shape of the *"penguin"*, the fine geometric details of the *"vase"*, and the structure of the *"chair"*.

Implementation details. We train SHAP-EDITOR with a fixed noise level ($\sigma_\tau = 0.308$). We use IP2P [3] as $\hat{\epsilon}_{\text{TI2I}}$ for global editing. For local editing, we employ Stable Diffusion v1-5 [56] for $\hat{\epsilon}_{\text{T2I}}$ and MagicBrush [85] (a fine-tuned version of IP2P) for $\hat{\epsilon}_{\text{TI2I}}$. *All* 3D objects used for evaluation, including those in quantitative and qualitative results, are *"unseen"*, *i.e.*, not used to train the editor. This differs from previous methods that perform test-time optimisation. Further details are provided in the supp. mat.

4.2. Comparison to the state of the art

We compare our method to other text-driven 3D editors such as Instruct-NeRF2NeRF (IN2N) [17], Vox-E [60], and Text2Mesh [44]. Specifically, IN2N iteratively edits images rendered from a NeRF with a TI2I editor (IP2P) and uses the edited images to gradually update the NeRF. Vox-E optimises a grid-based representation [26] by distilling knowledge from a T2I model (Stable Diffusion) with volumetric regularisation; a refinement stage is added for localised edits. Text2Mesh optimises meshes with CLIP similarity between the mesh and the target prompt. Since different methods receive different input formats (NeRF, mesh, and voxel grid), we provide many (200) rendered images at 512 × 512 resolution for initialising their 3D representations.

We consider two variants of SHAP-EDITOR: **(i) Ours (Single-prompt):** SHAP-EDITOR trained with a single prompt at a time and multiple classes (this is the default setting in our experiments), and **(ii) Ours (Multi-prompt):** SHAP-EDITOR trained with multiple prompts and multiple classes. Finally, we also consider a test-time optimisation baseline (**Ours (Test-time Optimisation)**), where, instead of learning an editor function, the Shape-E latent is optimised directly to minimise the same set of losses.

Quantitative comparison. Table 1 compares methods quantitatively. Both the single-prompt and multi-prompt variants of our approach are superior to optimisation-based 3D editing methods, despite addressing a harder problem, *i.e.*, the test 3D assets are not seen during training. The inference of SHAP-EDITOR is near-instantaneous (within one second) since editing requires only a single forward pass.

Qualitative comparison. Figure 3 compares methods qualitatively. All prior works struggle with global edits. Text2Mesh results in noisy outputs and undesired structural changes. IN2N is able to preserve the shape and identity of the original objects but fails to converge for some prompts, such as *"Make its color look like rainbow"*. The reason is that edited images produced by IP2P under this prompt have almost no consistency and, as a result, they cannot be inte-

Figure 4. **Generalisation to unseen categories.** "Seen categories" refer to object classes included in the training dataset; the specific instances shown were not used for training. "Unseen categories" represent the object classes that were never encountered during training.

grated coherently into 3D. On the other hand, Vox-E successfully changes the appearance of the objects, but due to distillation from a T2I model rather than a TI2I model, it fails to preserve the geometry of the objects.

When local edits are desired, such as *"Add a Santa hat to it"* (Figure 3, bottom row), Text2Mesh and IN2N do not produce meaningful changes. Text2Mesh mainly changes the texture, and IN2N ignores the instruction entirely. This can be attributed to the inability of their underlying 2D models to add or remove objects. Vox-E adds the hat to the penguin, but other regions (*e.g.*, nose) also change unintentionally, despite their spatial refinement stage.

The combination of training objectives in our approach leverages the complementary aspects of different 2D diffusion priors, overcoming these problems even while using feed-forward prediction. Furthermore, the learned editor also improves over test-time optimisation results with the same prompt and optimisation objectives. We hypothesise that this is because learning an editor can regularise the editing process too. Finally, while a single-prompt editor achieves the best results, we show that it is possible to train an editor with multiple prompts (last column) without significantly compromising fidelity or structure.

Figure 4 provides additional results for various instructions, each associated with a single-prompt editor. Our trained editors are capable of performing consistent edits across diverse objects, and, importantly, generalise to *unseen categories* not included in the training dataset.

4.3. Ablation study

Quantitative analysis. Table 2a presents the quantitative results for global editing, where the omission of depth regularisation leads to a noticeable degradation in performance, reflected by high Structure Dist. Likewise, the removal of loss components for local editing impairs the model to varying extents (Table 2b), which we analyse next.

Model	CLIP$_{sim}$ ↑	CLIP$_{dir}$ ↑	Structure Dist. ↓
Ours w/o $\mathcal{L}_{\text{reg-global}}$	0.218	0.058	0.138
Ours	**0.272**	**0.070**	**0.008**

(a) Ablation study for global editing.

Model	CLIP$_{sim}$ ↑	CLIP$_{dir}$ ↑
Ours w/o cross-attn masks	0.261	0.064
Ours w/o $\mathcal{L}_{\text{reg-local}}$	0.282	0.092
Ours w/o $\mathcal{L}_{\text{SDS-T2I}}$	0.263	0.067
Ours w/o $\mathcal{L}_{\text{SDS-TI2I}}$	0.278	0.096
Ours	**0.292**	**0.097**

(b) Ablation study for local editing.

Table 2. **Quantitative ablation study on loss components.**

Qualitative analysis. In Figure 5, we illustrate the effect of the different model components. For global editing, eliminating the depth regularisation term (*i.e.*, Ours w/o $\mathcal{L}_{\text{reg-global}}$) can lead to significant alterations of the source shape. For local editing, we observe the following: (i) the **cross-attn masks** specify the editable region where regularisation is not applied. If such a region is not defined, the depth and photometric regularisers would be applied to the whole object, thereby forbidding the formation of local shapes (in this case, the *Santa hat*); (ii) the **regularisation loss** ($\mathcal{L}_{\text{reg-local}}$) helps the model to maintain the object's identity (both appearance and shape); (iii) the **T2I loss** ($\mathcal{L}_{\text{SDS-T2I}}$) significantly improves the quality of local editing. When omitted (*i.e.*, Ours w/o $\mathcal{L}_{\text{SDS-T2I}}$), only the TI2I

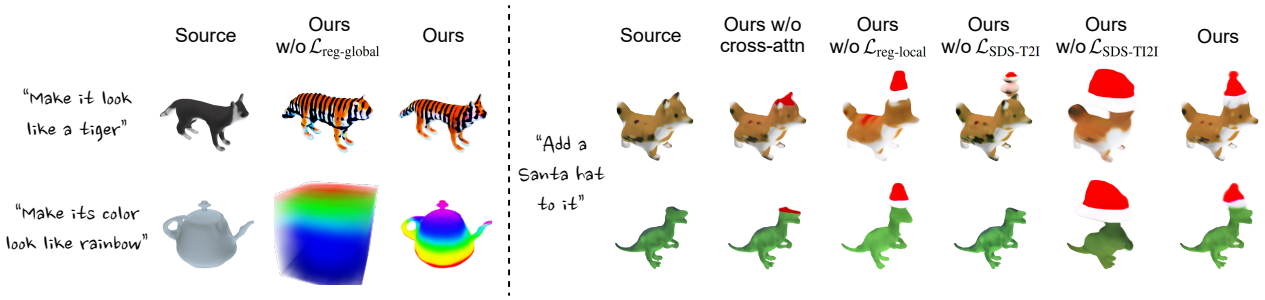

Figure 5. **Qualitative ablation results,** where the left and right parts correspond to global and local editing, respectively.

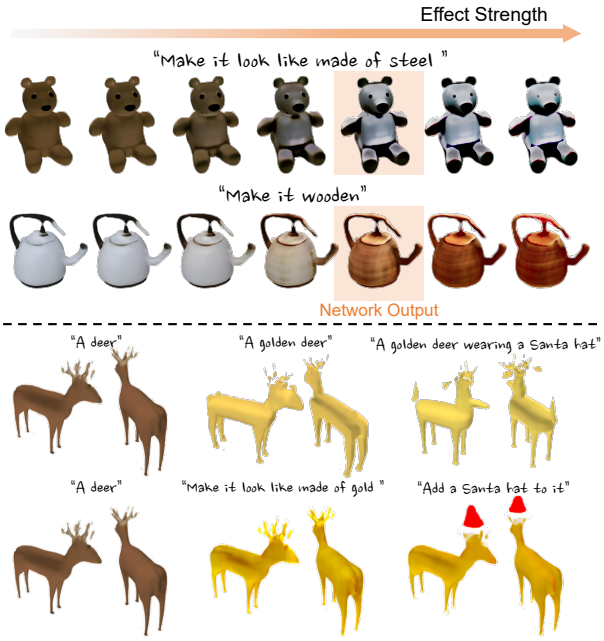

Figure 6. **Top**: the strength of the editing effects can be controlled via linear interpolation and extrapolation in latent space. **Bottom**: the examples in the first row are directly generated by Shap-E and the second row is generated by progressively adding multiple effects to the unseen category *"deer"*.

prior is used, which struggles with localised edits (same issues that [17, 24] exhibit); (iv) the **TI2I loss** ($\mathcal{L}_{\text{SDS-TI2I}}$) uses source images as references, which greatly helps with understanding the layout of edits. Thus, Ours w/o $\mathcal{L}_{\text{SDS-TI2I}}$ leads to spatial inaccuracy in editing (same as [59]).

4.4. Discussion

In Figure 6 (top), we observe that the latent space of Shap-E is partially linear. After training the editor to produce the desired effects, we can further control the strength of the effects. This could be done by scaling the residual between the updated latent and source latent by a factor η. The editor's output corresponds to $\eta = 1$. Increasing (decreasing) η weakens (strengthens) the effects. In Figure 6 (bottom), we show that edits can be accumulated progressively until the desired effect is achieved. Furthermore, as noted in [23]

Figure 7. **Unified editing vector.** The editing effects can be transferred via simple vector arithmetic operations in latent space.

and shown in the figure, Shap-E (the first row of the bottom part) itself fails at compositional object generation, but our approach can largely remedy that by decomposing complex prompts into a series of edits. Finally, in Figure 7, we also show that some of the edits, once expressed in latent space, are quite linear. By this, we mean that we can find a single vector for effects like *"Make its color look like rainbow"* or *"Turn it into pink"* that can be used to edit any object by mere addition regardless of the input latent. This is a strong indication that the latent space is well structured and useful for semantic tasks like editing. For general remarks on the research ethics, please see the supp. mat.

Limitations. Our work is based on the latent space of Shap-E and pre-trained 2D editors, which pose an upper bound on quality and performance. Further, while we show that we can learn a latent editor that understands multiple instructions, we could not yet achieve a fully open-ended editor. We conjecture that this might require training at a much larger scale than we can afford (*i.e.*, hundreds of GPUs).

5. Conclusion

We have introduced SHAP-EDITOR, a 3D editor that operates in latent space. It eschews costly test-time optimisation and runs in a feed-forward fashion within one second for any object. SHAP-EDITOR is trained from multiple 2D diffusion priors and thus combines their strengths, achieving compelling results for both global and local edits, even when compared to slower optimisation-based 3D editors.

Acknowledgements. This research is supported by ERC-CoG UNION 101001212 and EPSRC VisualAI EP/T028572/1.

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. In *CVPR*, pages 20919–20929, 2023. 2

[2] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. In *ECCV*, 2022. 2

[3] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, pages 18392–18402, 2023. 1, 2, 3, 5, 6

[4] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. In *SIGGRAPH*, 2023. 2

[5] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. Neuraleditor: Editing neural radiance fields via manipulating point clouds. In *CVPR*, pages 12439–12448, 2023. 2

[6] Minghao Chen, Iro Laina, and Andrea Vedaldi. Training-free layout control with cross-attention guidance. In *WACV*, 2023. 2, 5

[7] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023. 2

[8] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *ECCV*, pages 170–187. Springer, 2022. 2

[9] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3d: Progressively local editing for text-to-3d content creation with complex semantic prompts. *arXiv preprint arXiv:2310.11784*, 2023. 3

[10] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. In *NeurIPS*, 2023. 2, 5

[11] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model. *NeurIPS*, 35:8882–8895, 2022. 2

[12] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. In *SIGGRAPH*, 2021. 5

[13] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *ICLR*, 2023. 2

[14] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *NeurIPS*, 35:31841–31854, 2022. 2

[15] Bingchen Gong, Yuehao Wang, Xiaoguang Han, and Qi Dou. Recolornerf: Layer decomposed radiance field for efficient color editing of 3d scenes. *arXiv preprint arXiv:2301.07958*, 2023. 2

[16] Ori Gordon, Omri Avrahami, and Dani Lischinski. Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. *ICCV*, 2023. 2

[17] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *ICCV*, 2023. 1, 3, 6, 8

[18] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *ICLR*, 2023. 1, 2

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS*, 2021. 5

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 1, 2, 3

[21] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *CVPR*, pages 867–876, 2022. 2

[22] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 2

[23] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 1, 2, 3, 8

[24] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*, 2023. 1, 3, 8

[25] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. Conerf: Controllable neural radiance fields. In *CVPR*, pages 18623–18632, 2022. 2

[26] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 6

[27] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, 2023. 2

[28] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *NeurIPS*, 35:23311–23330, 2022. 2

[29] Adam R Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Jimenez Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *ICML*, pages 5742–5752. PMLR, 2021. 2

[30] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. Palettenerf: Palette-based appearance editing of neural radiance fields. In *CVPR*, pages 20691–20700, 2023. 2

[31] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *CVPR*, 2023. 2

[32] Jae-Hyeok Lee and Dae-Shik Kim. Ice-nerf: Interactive color editing of nerfs via decomposition-aware weight optimization. In *ICCV*, pages 3491–3501, 2023. 2

[33] Jiabao Lei, Yabin Zhang, Kui Jia, et al. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *NeurIPS*, 35:30923–30936, 2022. 2

[34] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022. 2

[35] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *CVPR*, pages 10965–10975, 2022. 2

[36] Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3d editing via focal-fusion assembly. *arXiv preprint arXiv:2308.10608*, 2023. 3, 5

[37] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. Gligen: Open-set grounded text-to-image generation. In *CVPR*, 2023. 2

[38] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, pages 300–309, 2023. 2

[39] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *ICCV*, pages 5773–5783, 2021. 2

[40] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. ATT3D: amortized text-to-3D object synthesis. In *ICCV*, 2023. 2

[41] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *CVPR*, pages 7086–7096, 2022. 2

[42] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022. 2

[43] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *CVPR*, pages 12663–12673, 2023. 2

[44] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *CVPR*, pages 13492–13502, 2022. 2, 6

[45] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *ICCV*, pages 14607–14619, 2023. 2

[46] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[47] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *CVPR*, pages 20669–20679, 2023. 2

[48] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 2

[49] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *CVPR*, pages 6038–6047, 2023. 1, 2

[50] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2

[51] Jangho Park, Gihyun Kwon, and Jong Chul Ye. Ed-nerf: Efficient text-guided editing of 3d scene using latent space nerf. *arXiv preprint arXiv:2310.02712*, 2023. 3

[52] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *SIGGRAPH*, 2023. 2

[53] Yicong Peng, Yichao Yan, Shengqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation. *NeurIPS*, 35:31402–31415, 2022. 2

[54] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2, 3

[55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 2

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 1, 6

[57] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 2

[58] Aditya Sanghi, Rao Fu, Vivian Liu, Karl DD Willis, Hooman Shayani, Amir H Khasahmadi, Srinath Sridhar, and Daniel Ritchie. Clip-sculptor: Zero-shot generation of high-fidelity and diverse shapes from natural language. In *CVPR*, pages 18339–18348, 2023. 2

[59] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *ICCV*, 2023. 1, 3, 5, 6, 8

[60] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *ICCV*, pages 430–440, 2023. 6

[61] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid represen-

tation for high-resolution 3d shape synthesis. *NeurIPS*, 34: 6087–6101, 2021. 2

[62] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265. PMLR, 2015. 1

[63] Hyeonseop Song, Seokhun Choi, Hoseok Do, Chul Lee, and Taehyeong Kim. Blending-nerf: Text-driven localized editing in neural radiance fields. In *ICCV*, pages 14383–14393, 2023. 2

[64] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1

[65] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022. 2

[66] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022. 2

[67] Narek Tumanyan, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Splicing vit features for semantic appearance transfer. In *CVPR*, 2022. 5

[68] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023. 2

[69] Binglun Wang, Niladri Shekhar Dutt, and Niloy J Mitra. Proteusnerf: Fast lightweight nerf editing using 3d-aware image context. *arXiv preprint arXiv:2310.09965*, 2023. 3

[70] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *CVPR*, pages 3835–3844, 2022. 2

[71] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE TVCG*, 2023. 2

[72] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. *arXiv preprint arXiv:2305.15094*, 2023. 2

[73] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 2

[74] Silvan Weder, Guillermo Garcia-Hernando, Aron Monszpart, Marc Pollefeys, Gabriel J Brostow, Michael Firman, and Sara Vicente. Removing objects from neural radiance fields. In *CVPR*, pages 16528–16538, 2023. 2

[75] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *CVPR*, 2023. 5

[76] Shiyao Xu, Lingzhi Li, Li Shen, and Zhouhui Lian. Desrf: Deformable stylized radiance field. In *CVPR*, pages 709–718, 2023. 2

[77] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *ECCV*, pages 159–175. Springer, 2022. 2

[78] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, pages 13779–13788, 2021. 2

[79] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *ECCV*, pages 597–614. Springer, 2022. 2

[80] Lu Yu, Wei Xiang, and Kang Han. Edit-diffnerf: Editing 3d neural radiance fields using 2d diffusion model. *arXiv preprint arXiv:2306.09551*, 2023. 3

[81] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *CVPR*, pages 18353–18364, 2022. 2

[82] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022. 2

[83] Hao Zhang, Yao Feng, Peter Kulits, Yandong Wen, Justus Thies, and Michael J Black. Text-guided generation and editing of compositional 3d avatars. *arXiv preprint arXiv:2309.07125*, 2023. 3

[84] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, pages 717–733. Springer, 2022. 2

[85] Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023. 2, 6

[86] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 2

[87] Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, Caiming Xiong, and Ran Xu. Hive: Harnessing human feedback for instructional visual editing. *arXiv preprint arXiv:2303.09618*, 2023. 2

[88] Chengwei Zheng, Wenbin Lin, and Feng Xu. EditableNeRF: Editing topologically varying neural radiance fields by key points. In *CVPR*, 2023. 2

[89] Xingchen Zhou, Ying He, F Richard Yu, Jianqiang Li, and You Li. Repaint-nerf: Nerf editting via semantic masks and diffusion models. *arXiv preprint arXiv:2306.05668*, 2023. 3

[90] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. *arXiv preprint arXiv:2306.13455*, 2023. 3