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

Supplementary Material

A. Overview

This supplementary material contains the following parts:

- **Implementation details (Appendix B).** We provide full details regarding the dataset formulation and experimental settings.
- **Additional results (Appendix C).** We provide additional examples of our method, including additional global and local prompts, results on human avatar editing, and we further demonstrate the potential of our multi-prompt editor to handle a large number of prompts.
- **Additional ablation study (Appendix D).** We provide additional ablation studies of SHAP-EDITOR on the initialisation method, the choice of σ_τ, and the attention maps used to guide the regularisation loss for local editing.
- **Extended discussion on prior methods (Appendix E).** We discuss the difference between our SHAP-EDITOR and other related 3D editing methods.
- **Failure cases (Appendix F).** We provide failure cases of our method qualitatively.
- **Ethics (Appendix G).** We provide ethics discussion of data used in our paper.

B. Implementation details

B.1. Dataset formulation

In this section, we provide more details regarding the construction of the training and evaluation datasets.

Training dataset. The training dataset specifies a set of 3D objects used to train different instructions. There are in total 33 object classes, each containing up to 10 instances.

Specifically, we use Shap-E-generated objects spanning 20 object classes: *apple, banana, candle, cat, chair, corgi, dinosaur, doctor, duck, guitar, horse, microphone, penguin, pineapple, policeman, robot, teapot, teddy bear, toy plane, vase*

In addition, we use 3D objects from OmniObject3D [11] spanning 21 object classes: *bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, guitar, hippo, mouse, panda, pineapple, rabbit, rhino, scissor, teapot, teddy bear, toy plane, vase, zebra*

Note that, we manually filter out the invalid instruction-instance pairs during training. For example, we consider it unreasonable to *"add a Santa hat"* to *"chairs"*, thereby discarding such pairs. Consequently, we obtain a set of valid object classes for each editing instruction during training, as summarised in Tab. A1.

Evaluation dataset. The evaluation dataset consists of 20 high-quality instance-instruction pairs (12 global editing pairs and 8 local editing pairs), with the details listed in Table A2. In summary, there are 3 and 2 editing instructions for global and local editing, respectively, with 8 Shap-E generated objects and 7 instances sourced from OmniObject3D. Note that *none* of the instances in the evaluation dataset are utilised for training purposes.

B.2. Experimental details

Shap-E settings. The encoder h takes as input an RGB point cloud (16384 points) and different views (20) of the 3D asset from random camera angles at 256×256 resolution. The outputs of the encoder are latents with shape 1024×1024.

The decoder outputs the parameters of a neural field represented as a 6-layer MLP. The weights of the first four layers are linear transformations of the latent, while the weights of the last two layers are fixed. The output feature vector computed through the MLP is then mapped to the neural field's density and RGB values (or alternatively, SDF and texture color) using different heads.

Finally, Shap-E uses a generative latent-space model for which it employs a transformer-based diffusion architecture akin to Point-E [8], with latent dimensions of 1024×1024. It offers two pre-trained conditional diffusion models: image-conditional and text-conditional. The image-conditional approach, paralleling Point-E, augments the transformer context with a 256-token CLIP embedding. The text-conditional model introduces a single token to the transformer context. We use the text-conditional model in our paper.

SDS with classifier guidance. During the Score Distillation Sampling (SDS) process, we adopt the classifier-free guidance [4] to enhance the signal of each underlying 2D model for distillation purposes. Specifically, for the text-guided image-to-image (TI2I) SDS, we define:

$$\begin{aligned}\hat{\boldsymbol{\epsilon}}^*_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{x}^s, y, t) = \hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{\varnothing}, \varnothing, t) \\ + \gamma_I \cdot \big(\hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{x}^s, \varnothing, t) - \hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{\varnothing}, \varnothing, t)\big) \\ + \gamma_T \cdot \big(\hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{x}^s, y, t) - \hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{x}^s, \varnothing, t)\big), \quad (5)\end{aligned}$$

where γ_I and γ_T correspond to image and text guidance scales, respectively. Then:

$$\nabla_{\boldsymbol{x}_e} \mathcal{L}_{\text{SDS-TI2I}}(\boldsymbol{x}^e \mid \boldsymbol{x}^s, y) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\Big[\hat{\boldsymbol{\epsilon}}^*_{\text{TI2I}}(\boldsymbol{x}^e_t; \boldsymbol{x}^s, y, t) - \boldsymbol{\epsilon}\Big] \quad (6)$$

Editing type	Instruction	Object class
Global	*"Make it look like made of gold"*	*apple, banana, candle, cat, chair, corgi, dinosaur, doctor, duck, guitar, horse, microphone, penguin, pineapple, policeman, robot, teapot, teddy bear, toy plane, vase;* ***bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, guitar, hippo, mouse, panda, pineapple, rabbit, rhino, scissor, teapot, teddy bear, toy plane, vase, zebra***
Global	*"Make it look like a tiger"*	*cat, corgi, dinosaur, duck, horse, penguin, teddy bear;* ***bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, hippo, mouse, panda, rabbit, rhino, teddy bear, zebra***
Global	*"Make its color look like rainbow"*	*apple, banana, candle, cat, chair, corgi, dinosaur, doctor, duck, guitar, horse, microphone, penguin, pineapple, policeman, robot, teapot, teddy bear, toy plane, vase;* ***bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, guitar, hippo, mouse, panda, pineapple, rabbit, rhino, scissor, teapot, teddy bear, toy plane, vase, zebra***
Local	*"Add a Santa hat to it"*	*cat, corgi, dinosaur, doctor, duck, horse, penguin, policeman, teddy bear;* ***bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, hippo, mouse, panda, rabbit, rhino, teddy bear, zebra***
Local	*"Make it wear a blue sweater"*	*cat, corgi, dinosaur, doctor, duck, horse, penguin, policeman, teddy bear;* ***bear, cat, cow, dinosaur, dog, duck, elephant, giraffe, hippo, mouse, panda, rabbit, rhino, teddy bear, zebra***

Table A1. **Training dataset formulation.** The object classes in ***bold*** are sourced from OmniObject3D, whereas the remaining classes are generated from text prompts using Shap-E.

Editing type	Instruction	Instance
Global	*"Make it look like made of gold"*	*"A bird",* ***"An apple"****,* ***"A scissor"****,* ***"A vase"***
	"Make it look like a tiger"	*"A cat", "A corgi",* ***"A dinosaur"****,* ***"A zebra"***
	"Make its color look like rainbow"	*"A chair", "A teapot",* ***"A guitar"****,* ***"A pineapple"***
Local	*"Add a Santa hat to it"*	*"A corgi", "A penguin", "A robot",* ***"A dinosaur"****,* ***"A teddy bear"***
	"Make it wear a blue sweater"	*"A corgi", "A penguin",* ***"A teddy bear"***

Table A2. **Evaluation dataset formulation.** The instances in ***bold*** are sourced from OmniObject3D, whereas the remaining instances are generated from text prompts using Shap-E. The specific object instances used for evaluation are not seen during training.

Similarly, for text-to-image (T2I) SDS,

$$\hat{\boldsymbol{\epsilon}}^{*}_{\text{T2I}}(\boldsymbol{x}^e_t; y^e, t) = \hat{\boldsymbol{\epsilon}}_{\text{T2I}}(\boldsymbol{x}^e_t; \varnothing, t) + \gamma'_T \cdot \big(\hat{\boldsymbol{\epsilon}}_{\text{T2I}}(\boldsymbol{x}^e_t; y^e, t) - \hat{\boldsymbol{\epsilon}}_{\text{T2I}}(\boldsymbol{x}^e_t; \varnothing, t)\big), \quad (7)$$

where γ'_T denotes the text guidance scale, and

$$\nabla_{\boldsymbol{x}_e} \mathcal{L}_{\text{SDS-T2I}}(\boldsymbol{x}^e \,|\, y^e) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\Big[\hat{\boldsymbol{\epsilon}}^{*}_{\text{T2I}}(\boldsymbol{x}^e_t; y^e, t) - \boldsymbol{\epsilon}\Big] \quad (8)$$

For global editing, where only TI2I SDS is applied, we consider a default setting of guidance scales $(\gamma_I, \gamma_T) = (2.5, 50)$. For local editing, we adopt the guidance scales $(\gamma_I, \gamma_T, \gamma'_T) = (2.5, 7.5, 50)$.

Loss configuration. In terms of the overall loss for global editing, we consider a weighted combination of TI2I and global regularisation losses,

$$\mathcal{L}_{\text{global}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e) = \lambda_{\text{TI2I}} \cdot \mathcal{L}_{\text{SDS-TI2I}}(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y) + \lambda_{\text{reg-global}} \cdot \mathcal{L}_{\text{reg-global}}(\boldsymbol{d}^e, \boldsymbol{d}^s), \quad (9)$$

with loss scales indicated by λ_{TI2I} and $\lambda_{\text{reg-global}}$, respectively.

For local editing, we use a weighted combination of the TI2I, T2I, and local regularisation losses:

$$\mathcal{L}_{\text{local}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e, \boldsymbol{m}) = \lambda_{\text{TI2I}} \cdot \mathcal{L}_{\text{SDS-TI2I}}(\boldsymbol{x}^e \,|\, \boldsymbol{x}^s, y) + \lambda_{\text{T2I}} \cdot \mathcal{L}_{\text{SDS-T2I}}(\boldsymbol{x}^e \,|\, y^e) + \mathcal{L}_{\text{reg-local}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e, \boldsymbol{m}), \quad (10)$$

where λ_{TI2I} and λ_{T2I} denote corresponding loss scales, and $\mathcal{L}_{\text{reg-local}}(\boldsymbol{x}^s, \boldsymbol{x}^e, \boldsymbol{d}^s, \boldsymbol{d}^e, \boldsymbol{m})$ is defined by Eq. 4 in the main text.

Estimation of local editing region. An estimate of local editing regions can be obtained by extracting the cross-attention maps from pre-trained 2D models (*i.e.*, MagicBrush). Specifically, given an example editing prompt *"Add a Santa hat to it"*, we first calculate the cross-attention maps between the image features and the word token *"hat"*. We then average all cross-attention maps corresponding to feature resolution 32×32 at a particular timestep $t = 600$. The averaged map undergoes a series of post-processing steps, including (i) bilinear upsampling to a higher resolution at 128×128; (ii) hard thresholding at 0.5; (iii) spatial dilation by 10 pixels; (iv) Gaussian blurring by 5 pixels. The final mask $\boldsymbol{m}$ is then adopted as an approximation of the editable region, and used in Eq. 4 in the main text.

Model settings. As mentioned previously, we consider two variants in our method, namely **Ours (Single-prompt)**

and **Ours (Multi-prompt)**. We train both methods on objects from the entire training dataset, to ensure their applicability across multiple instances of various categories.

In terms of the editing instructions, **Ours (Single-prompt)** is considered as our default model and designed to handle *one* prompt at one time. Consequently, it requires 5 independent models for each of the editing instructions in the evaluation dataset. In contrast, **Ours (Multi-prompt)** is trained on a combination of editing instructions and is capable of performing different edits according to the input text prompt. We train this multi-prompt model on all 5 instructions from the evaluation dataset simultaneously.

Architecture details. For SHAP-EDITOR's architecture, we use a similar network architecture as the text-conditional diffusion model in Shap-E [6], a transformer-based network. The original Shap-E text-to-3D network takes a noisy latent $\sigma_\tau \boldsymbol{r}^s + \alpha_\tau \boldsymbol{\epsilon}$ as input and directly predicts the original clean latent $\boldsymbol{r}^s$. Instead, the goal of our editor is to transform the original latent $\boldsymbol{r}^s$ to an edited one. Therefore, to support $(\sigma_\tau \boldsymbol{r}^s + \alpha_\tau \boldsymbol{\epsilon}, \boldsymbol{r}^s)$ as our input, we add additional input channels to the first linear projection layer. All weights are initialised by the weights of the pre-trained Shap-E text-to-3D diffusion model, while the weights that apply to the additional input channels are initialized to zeros following a similar setting to [1].

Rendering details. During the training phase, camera positions are randomly sampled using a circular track. This track has a radius of 4 units and a constant elevation angle of 30°. The azimuth angle varies within the range of $[-180°, 180°]$. For loss computation, images of the source NeRF and edited NeRF are rendered at a resolution of 128×128.

Training details. We adopt a constant learning rate of $1e^{-4}$, utilising the Adam optimiser with $\beta = (0.9, 0.999)$, a weight decay of $1e^{-2}$, and $\epsilon = 1e^{-8}$. The batch size is 64. We train our single-prompt and multi-prompt models for 150 and 500 epochs, respectively. We use the same timestep for $\hat{\boldsymbol{\epsilon}}_{\text{TI2I}}(\boldsymbol{x}_t^e; \boldsymbol{x}^s, y, t)$ and $\hat{\boldsymbol{\epsilon}}_{\text{T2I}}(\boldsymbol{x}_t^e; y^e, t)$, which is randomly sampled from $[0.02, 0.98]$ following the setting in [9]. We also adopt an annealing schedule for the max timestep. After 100 and 300 epochs for the single- and multi-prompt versions, respectively, the max timestep of t decreases with a ratio of 0.8 for every 10 epochs and 50 epochs. The annealing scheduler helps the model capture more details in the late training.

We set $\lambda_{\text{T2I}} = \lambda_{\text{TI2I}} = 1$, and the regularisation terms $\lambda_{\text{photo}} = 1.25$ and $\lambda_{\text{depth}} = 0.8$ for local editing and $\lambda_{\text{reg-global}} = 5$ for global editing in order to better preserve structure. We also employ a linear warmup schedule for the photometric loss in the early epochs. This helps the model to first focus on generating correct semantics, such as *"a penguin wearing a Santa hat"*, and then gradually reconstructing the appearance and shape of the original object (*e.g.*, the *"penguin"*) with the help of masked loss, *i.e.* recovering the identity of the original object. The training of each single-prompt model takes approximately 10 GPU hours, and the multi-prompt model takes 30 GPU hours, on the NVIDIA RTX A6000.

Evaluation details. During evaluation, to compute the CLIP metrics and the Structure Distance, we uniformly sample 20 viewpoints following the same recipe as in the training phase. All rendered images are resized to 256×256 to ensure a fair comparison across different methods.

C. Additional Results

Additional visualisations. Figure A1 provides additional visualised results for our SHAP-EDITOR, with each editing prompt associated with a distinct model, *i.e.*, Ours (Single-prompt). It is evident that our method is capable of performing accurate edits across diverse object classes and demonstrates reasonable generalisation to unseen categories.

Human Avatar Editing. Figure A2 shows additional semantic editing result with human avatars collected from Objaverse [2]. Our SHAP-EDITOR successfully change the style or transform the character into *"Clown"*, *"Tolkien Elf"* or *"Van Gogh"*. Although, we *can* achieve different semantic edits on human avatars from Objaverse, the quality of the 3D assets is still limited by the Shap-E latent space. More research is required for the latter to be on par with test-time instance optimization, but latent space editing is the key to computationally practical methods (1 s inference). Even so, our research shows the *potential* of this class of methods for editing.

Scaling up to more prompts. We further explore the possibility of learning more prompts within one editor model. We first train a 10-prompt model using the five instructions included in the original dataset plus extra five prompts: *"Make it look like a statue"*, *"Make it look like made of steel"*, *"Make it look like made of lego"*, *"Add a party hat to it"*, *"Add rollerskates to it"*. We also expand the instructions to train a 20-prompt model (which includes the previous 10 prompts plus *"Make it look like a panda"*, *"Make it look like made of bronze"*, *"Turn it into blue"*, *"Turn it into yellow"*, *"Turn it into pink"*, *"Turn it into green"*, *"Turn it into red"*, *"Add a snowboard to it"*, *"Add sunglasses to it"*, *"Add a crown to it"*). As shown in Fig. A3, the performance decreases slightly when moving from a single prompt to more prompts. However, the difference between 10 prompts and 20 prompts is marginal. This indicates the potential of our SHAP-EDITOR to scale to tens of prompts and even arbitrary prompts as inputs.

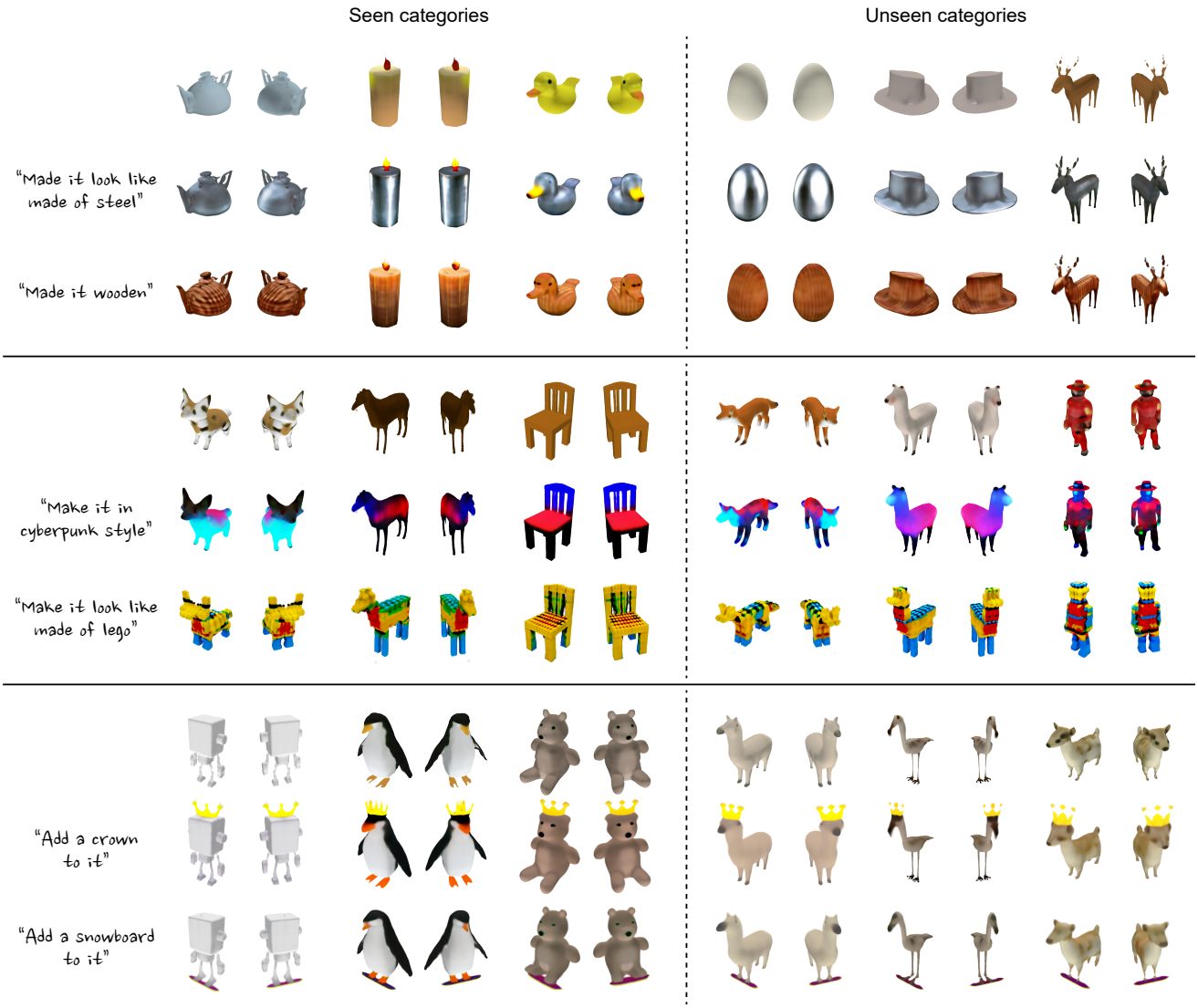

Figure A1. **Additional visualisations.** We apply different editing instructions (including both global and local edits) across various instances, also demonstrating the generalisability of our method to multiple unseen categories.

Figure A2. **Human avatar editing.** We apply SHAP-EDITOR for editing human avatars. Our SHAP-EDITOR successfully change the style or appearance of the provided avatars.

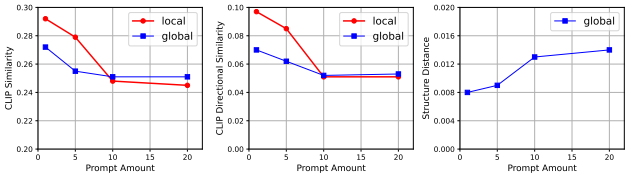

Figure A3. **Scale up the amount of prompts.** We explore the possibility of learning a single editor function with tens of prompts. As the amount of prompts increases, the CLIP similarly and CLIP directional similarity scores decrease. However, both scores reach a plateau when more prompts are introduced.

D. Additional ablation studies

Network initialisation. We compare the editing results of SHAP-EDITOR *with* and *without* the pre-trained weights of the text-to-3D Shap-E diffusion network quantitatively and qualitatively under the multi-prompt setting, *i.e.*, Ours (Multi-prompt). As shown in Fig. A4, the editor trained with Shap-E initialisation can successfully generate different effects given different prompts. However, if instead we randomly initialise the network, the editor is unsuccessful, yielding similar outputs regardless of the prompt. We assume this is because the model initialised with Shap-E pre-trained weights inherits its partial abilities to understand the natural language, and the randomly initialised one reaches a local optimum, ignoring the textual instructions.

Effects of σ_τ. Next, we study the value σ_τ that is used when noising the source latent to be able to initialise the editor with the pre-trained weights of the Shap-E diffusion model. To keep the full information and details of the original 3D asset, we follow [1] and concatenate the noised latent with the original latent $(\sigma_\tau \boldsymbol{r}^s + \alpha_\tau \boldsymbol{\epsilon}, \boldsymbol{r}^s)$. Here, σ_τ is a hyperparameter that controls the information we keep from the original latent in the noised counterpart. A smaller σ_τ means we keep less information. The σ_τ value in the main text corresponds to $\tau = 200$ in the original total 1024 Shap-E diffusion steps. A higher τ corresponds to a smaller σ_τ, and when $\tau = 1024$, the noised input can be considered as a random Gaussian noise. As illustrated in the Fig. A5, time steps in the range $[200, 600]$ result in only marginal differences in performance. A large noise or no noise will lead to a drop in the CLIP similarity and CLIP directional similarity scores. Increasing the noise also leads to a larger Structure Distance.

Choice of cross-attention maps. To accurately estimate the local editing region, we assess the quality of cross-attention maps extracted at various timesteps from several pre-trained 2D models, including InstructPix2Pix [1], MagicBrush [12], and Stable Diffusion v1.5 [5]. As demonstrated in Fig. A6, most cross-attention maps either fail to disentangle the region of interest from the main object or suffer from excessive background noise. In contrast, the cross-attention map extracted from MagicBrush at $t = 600$ (indicated by red boxes) effectively highlights the region associated with the attention word tokens (*i.e.*, *"hat", "sweater"*). Therefore, we adopt this setting as the default in our experiments.

E. Extended discussion on prior methods

Our method differs from prior work employing test-time optimisation techniques in two main ways: (i) We use a direct, feed-forward approach to function on the 3D (latent) representation, unlike others that gradually optimise the 3D

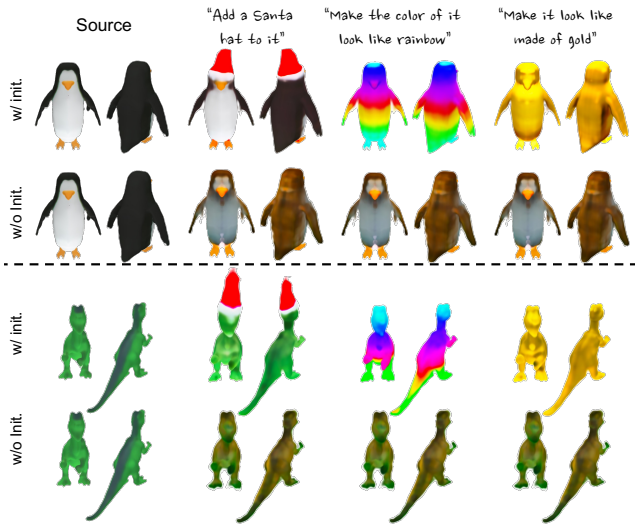

Figure A4. **Comparison with different initialisation method.** We use the multi-prompt setting in our experiments. "w/ init." denotes initialisation using the pre-trained weights of the Shap-E text-to-3D diffusion model, and "w/o init." indicates random initialisation. With random initialisation, the network loses the ability to distinguish across different prompts and produces similar results despite different instructions.

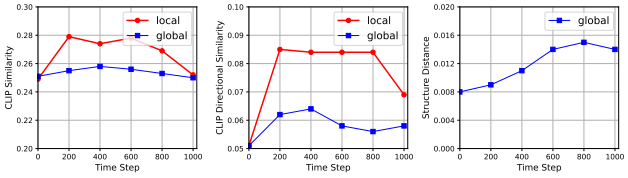

Figure A5. **Ablation study on the timestep τ for σ_τ.** We analyse the level of noise introduced to the input, with a large timestep τ corresponding to a large noise. As the timestep τ, thereby the noise level, increases, the Structure Distance rises consistently. When the input is the original latent ($\tau = 0$) or has very large noise ($\tau \to 1024$), we observe degraded performance in both CLIP similarity score and CLIP directional similarity score.

representation at test time to align it with a 2D prior. Our approach significantly reduces the inference time from tens of minutes to less than one second; (ii) Our method learns editing in a simpler, more structured latent space, avoiding the complexity of spaces like NeRF's weight matrix space. This simplification reduces learning difficulty and cost, allowing our model to generalise to novel objects at test time. Recently, EDNeRF [13] tries to edit NeRFs that are trained on the latent space of Stable Diffusion [5]. The loss changes from the image space to the VAE latent space of Stable Diffusion. In this context, the use of the term "latent" is different from ours since it still requires NeRF as a representation of the 3D model and test-time optimisation.

Another key difference in our work is the use of com-

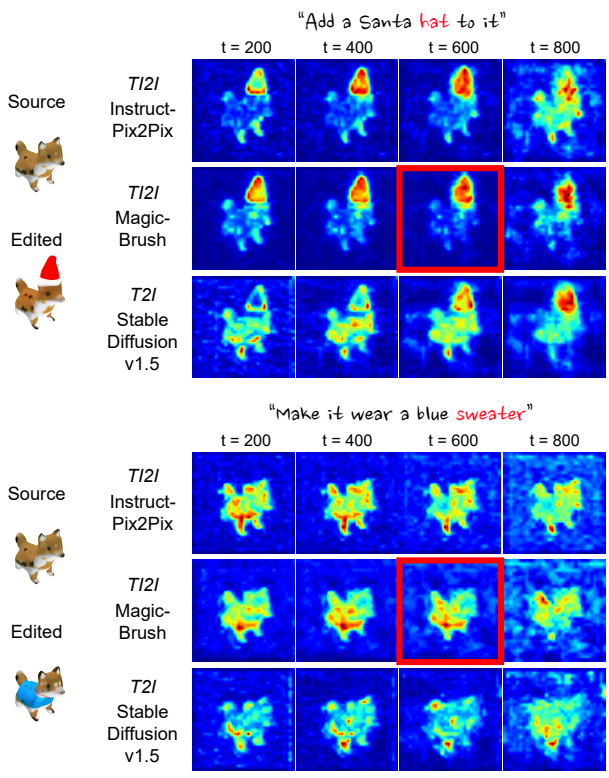

Figure A6. **Visualisation of cross-attention maps** that correspond to particular attention word tokens (labeled in red). These maps are extracted at different timesteps ($t \in [0, 1000]$) from various pre-trained 2D models (including InstructPix2Pix, MagicBrush, and Stable Diffusion v1.5). In this work, we consider the default cross-attention maps at $t = 600$ from MagicBrush, which are indicated by the red boxes.

plementary 2D diffusion priors for the training objectives. Other methods, such as IN2N [3], Vox-E [10], Instruct 3D-to-3D [7] typically distill knowledge from *one* network (*e.g.*, Stable Diffusion [5] for Vox-E [1] and InstructPix2Pix [1] for IN2N and Instruct 3D-to-3D) with different regularisation due to different 3D representations.

As shown in our ablation studies in the main text, distilling from only one network usually inherits the drawbacks of the original 2D model, such as the inability to edit locally or preserve the original appearance and structure. Instead, one can distill from multiple 2D editors to overcome these pitfalls and achieve better editing results.

Finally, we also experiment with the training objective of IN2N, *i.e.*, editing images directly and updating our editor function with a photometric loss. However, this led to divergence, likely due to the greater inconsistency introduced by training with multiple instances compared to optimising a single NeRF.

Figure A7. **Failure case.** When encountering a new class, such as *"turtle"*, which significantly differs from those in the training dataset, our model struggles to identify the correct position for local editing.

F. Failure case

In Fig. A7, we present two failure cases. These occur particularly when the model encounters an unseen class that significantly differs from the classes the editor was trained on. This disparity leads to difficulties in accurately determining the position for local editing, ultimately resulting in editing failures. We conjecture that such failure cases can be eliminated by training the editor on an even larger number of object categories.

G. Ethics

We use the OmniObject3D dataset and Objaverse following their terms and conditions. This data contains no personal data. For further details on ethics, data protection, and copyright please see https://www.robots.ox.ac.uk/~vedaldi/research/union/ethics.html.

References

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

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. 3

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

[4] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS*, 2021. 1

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 5, 6

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

[7] 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. 6

[8] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generat-

ing 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 1

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

[10] 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

[11] 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. 1

[12] Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *NeurIPS*, 2023. 5

[13] Chengwei Zheng, Wenbin Lin, and Feng Xu. EditableNeRF: Editing topologically varying neural radiance fields by key points. In *CVPR*, 2023. 5