View-Consistent 3D Editing with Gaussian Splatting

Yuxuan Wang[1], Xuanyu Yi[1,2], Zike Wu[1], Na Zhao[3], Long Chen[5], and Hanwang Zhang[1,4]

[1]Nanyang Technological University [2]Institute for Infocomm Research, A*STAR
[3]Singapore University of Technology and Design [4]Skywork AI
[5]Hong Kong University of Science and Technology

Appendix: Overview

In the supplementary material, we provide more details of our algorithm and the implementation in Sec. 1. Additionally, further discussion regarding the two Consistency Modules of VcEdit is available in Sec. 2. An extensive qualitative evaluation, including comparisons with NeRF-based editing methods, is conducted in Sec. 3. The settings of the user study are detailed in Sec. 4. Lastly, more visual results, extending the qualitative analysis of VcEdit, are presented in Sec. 5.

Algorithm 1 Detailed Pipeline of VcEdit in **One** Iteration

Input:
U-Net of Conditional Diffusion Model $\varepsilon_\theta(\cdot)$
Sequence of timesteps $T = t_1 > t_2 > \ldots > t_N = 1$
Source 3DGS $\mathcal{G}^{\text{src}}$
Source/target prompts as conditions y^{src}(e.g., "man"), y^{tgt} (e.g., "clown")

3DGS $\rightarrow$ **Images** (Input: $\mathcal{G}^{\text{src}} \rightarrow$ Output: $\mathcal{I}^{\text{src}}$)
  Rendering 3DGS to images: $\mathcal{I}^{\text{src}} = \mathcal{R}(\mathcal{G}^{\text{src}})$

Multi-view Images Editing (Input: $\mathcal{I}^{\text{src}} \rightarrow$ Output: $\mathcal{I}^{\text{edit}}$)
  Encoding images to latents: $z^{\text{src}}, z^{\text{ori}} = \mathcal{E}(\mathcal{I}^{\text{src}})$
  for $t = t_1, t_2, \ldots t_N$ **do**
    Sample noise: $\varepsilon \sim \mathcal{N}(0, \mathcal{I})$
    Add noise: $z_t = \sqrt{\alpha_t} z^{\text{src}} + \sqrt{1-\alpha_t}\varepsilon$
    Add noise to original latents: $z_t^{\text{ori}} = \sqrt{\alpha_t} z^{\text{ori}} + \sqrt{1-\alpha_t}\varepsilon$
    Noise prediction by **CCM**-applied U-Net: $\varepsilon_\theta, \varepsilon_\theta^{\text{ori}} = \varepsilon_\theta(z_t, y^{\text{tgt}}), \varepsilon_\theta(z_t^{\text{ori}}, y^{\text{src}})$
    Compute a noise offset proposed in [5]: $\Delta\varepsilon = (z_t^{\text{ori}} - \sqrt{\alpha_t} z^{\text{ori}})/\sqrt{1-\alpha_t}$
    Obtain the edited latents: $z^{\text{edit}} = (z^{\text{src}} - \sqrt{1-\alpha_t}(\varepsilon_\theta - \varepsilon_\theta^{\text{ori}} + \Delta\varepsilon))/\sqrt{\alpha_t}$
    Copy a 3DGS from source (**ECM** starts): $\mathcal{G}_{\text{ft}} = \text{copy}(\mathcal{G}^{\text{src}})$
    Fine-tune the 3DGS: $\mathcal{G}_{\text{ft}} = \underset{\mathcal{G}}{\text{argmin}} \sum_{v \in \mathcal{V}} \mathcal{L}_{\text{edit}}(\mathcal{R}(\mathcal{G}, v),\ \mathcal{D}(z_v^{\text{edit}}))$
    Render and encode to latents (**ECM** ends): $z^{\text{con}} = \mathcal{E}(\mathcal{R}(\mathcal{G}_{\text{ft}}, v)),\ v \in \mathcal{V}$
    Blend by the **CCM**'s consistent mask: $z^{\text{bld}} = \mathbf{M}^{\text{con}} * z^{\text{con}} + (1 - \mathbf{M}^{\text{con}}) * z^{\text{src}}$
    Next timestep: $z^{\text{src}} = z^{\text{bld}}$
  end for
  Decoding latents to edited guidance images: $\mathcal{I}^{\text{edit}} = \mathcal{D}(z^{\text{bld}})$

Images $\rightarrow$ **3DGS** (Input: $\mathcal{I}^{\text{edit}} \rightarrow$ Output: $\mathcal{G}^{\text{edit}}$)
  Fine-tune the source 3DGS: $\mathcal{G}^{\text{edit}} = \underset{\mathcal{G}}{\text{argmin}} \sum_{v \in \mathcal{V}} \mathcal{L}(\mathcal{R}(\mathcal{G}, v),\ \mathcal{I}^{\text{edit}})$

Output: The final edited 3DGS $\mathcal{G}^{\text{edit}}$.

1 More Details of our VcEdit

In the *Our Method* section of the main paper, we provide an overview of VcEdit along with the introductions of our two Consistency Modules. In this section, we provide more details in the algorithm and implementation of VcEdit with a step-by-step demonstration in Algorithm 1, where our two consistency modules are highlighted in BLUE color.

1.1 More Details in our Overall Pipeline

As outlined in the *Preliminary* and *Our Method* sections, VcEdit employs an iterative image-guided 3DGS editing pipeline that takes user-specified text prompt as instruction. Initially, the source 3DGS ($\mathcal{G}^{\text{src}}$) are rendered to images ($\mathcal{I}^{\text{src}}$) from various views (*line 1–2* in Algorithm 1). Subsequently, employing our specially designed multi-view image editing process based on InfEdit [5] (*line 3–18* in Algorithm 1), a set of multi-view consistent edited images ($\mathcal{I}^{\text{edit}}$) are generated and employed as image guidance to fine-tune $\mathcal{G}^{\text{src}}$ (*line 19–20* in Algorithm 1).

During fine-tuning, we adhere to the methodology proposed by the GSEditor [1], incorporating both a MAE loss and a VGG-based LPIPS loss [1, 3]. Additionally, we employ the HGS regularization, as suggested by GSEditor [1], which limits the positional shifts of the Gaussian induced by densification to maintain the essential information in $\mathcal{G}^{\text{src}}$. Consequently, the training objective is defined as:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{MAE}}(\mathcal{R}(\mathcal{G}, v), \mathcal{I}^{\text{edit}}) + \lambda \mathcal{L}_{\text{LPIPS}}(\mathcal{R}(\mathcal{G}, v), \mathcal{I}^{\text{edit}}) + \sum \lambda_j (P_j - \hat{P}_j) \quad (1)$$

Here, P_j indicates the original position of the j-th Gaussian in $\mathcal{G}$, while $\hat{P}_j$ denotes its shifted position during training. In VcEdit, we set the λ to 10 and each λ_i to 50. After 400 steps of fine-tuning, $\mathcal{G}^{\text{src}}$ transforms into the edited version, $\mathcal{G}^{\text{edit}}$. In subsequent iterations, $\mathcal{G}^{\text{edit}}$ becomes the new $\mathcal{G}^{\text{src}}$, and the editing cycle continues.

1.2 More Details in our Multi-view Images Editing Process

In the main paper, we simplified the explanation of our image editing procedure to enhance reader comprehension. This section elaborates on the process, integrating it with our baseline image editing framework, InfEdit [5] (*line 3–18* in Algorithm 1).

Initially, in *Our Method* section, we introduce our image editing process as a multi-timestep cycle, where each timestep's process is represented by $z^{\text{src}} \rightarrow z^{\text{edit}} \rightarrow z^{\text{con}} \rightarrow z^{\text{bld}}$. Following InfEdit [5], our detailed implementation introduces an additional set of **original latents**, z^{ori}, extracted from the original images (*line 4* in Algorithm 1). These latents are utilized by InfEdit in the add-noise and denoising process (*line 8–11* in Algorithm 1) and does not participate in any of our Consistency Module.

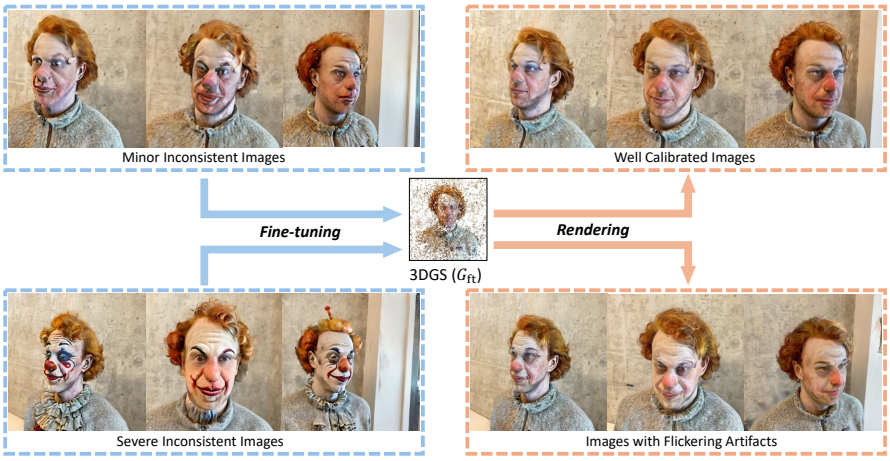

Fig. 1: Calibration Capability of 3DGS: A set of multi-view inconsistent images (*left*) serves as guidance for fine-tuning a 3DGS model, which is then re-rendered into images (*right*). The 3DGS model has the capability to calibrate minor inconsistency (*top*) but will exhibit flickering artifacts when trained with severe inconsistent images (*bottom*).

Moreover, InfEdit's U-Net architecture [5] is partitioned into three branches: the *Source Branch*, the *Layout Branch*, and the *Target Branch*. These branches are interconnected by two sets of cross-attention maps, which are denoted as $\mathbf{M} = \{\mathbf{M}^{\text{source-layout}}, \mathbf{M}^{\text{target}}\}$. Within our Cross-attention Consistency Module (CCM), we execute inverse rendering and subsequent re-rendering for both cross-attention map sets, ensuring uniform editing outcomes across all branches.

Our Editing Consistency Module is activated for every 5 timesteps for a more efficient forwarding. Since the z^{edit} in low-resolution does not satisfy the requirement of 3DGS fine-tuning, we initially decode them to images and use the images to fine-tune the 3DGS $\mathcal{G}_{\text{ft}}$ (*line 13* in Algorithm 1). After a rapid fine-tuning, we render the $\mathcal{G}_{\text{ft}}$ back to images, and encode the rendered images to obtain z^{con} (*line 14* in Algorithm 1).

2 Further Discussion on Consistency Modules

2.1 Discussion on Editing Consistency Module

In VcEdit, we introduce an innovative Editing Consistency Module (ECM) that leverages the subtle calibration potential of the 3DGS model to correct minor inconsistencies arising at each timestep of the image editing process. This section presents comparative experiments designed to further investigate the calibration capabilities and elucidate the effectiveness of our ECM.

In Fig. 1, we employ two sets of images, each exhibiting different levels of multi-view inconsistency, to guide the fine-tuning of a 3DGS. This model is then

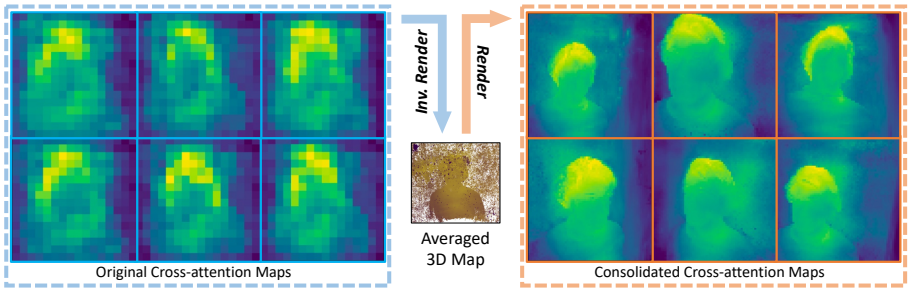

Fig. 2: Consolidation Effect of Our CCM on the "man → clown" sample: Initially, incoherent multi-view cross-attention maps (*left*) undergo an inverse rendering process into 3D space, resulting in an averaged 3D map. This 3D map is subsequently rendered back into the respective views, yielding consolidated cross-attention maps (*right*), which provides unified attention weights across various views to each 3D region.

rendered back into images (*right*). The *top* row illustrates the model's capability to correct minor multi-view inconsistencies effectively, as shown by the absence of mode collapse in the rendered images (*top-right*). In contrast, the *bottom* row reveals the model's limitations when confronted with severe inconsistencies, which lead to poorly calibrated images marked by flickering artifacts (*bottom-right*).

This observation elucidates the efficacy of our Editing Consistency Module in addressing inconsistencies by continually calibrating minor discrepancies at each image editing timestep, while directly employing original multi-view inconsistent images as guidance results in mode collapse.

2.2 Discussion on Cross-attention Consistency Module

In the main paper, we introduce the Cross-attention Consistency Module (CCM) as a novel approach to synchronize the attentive regions across all views within the U-Net layers, thereby facilitating the U-Net to produce multi-view consistent predictions. Specifically, the cross-attention maps from all views are consolidated through a process of inverse-rendering (2D → 3D) and subsequent rendering (3D → 2D).

Fig. 2 showcases the impact of our CCM on the "man → clown" example, comparing multi-view cross-attention maps towards the term "clown" before and after the consolidation process. The comparison reveals that, unlike the original cross-attention maps on the *left*, which are coarse and inconsistent across views, the consolidated maps exhibit a refined, view-consistent attention on each 3D region.

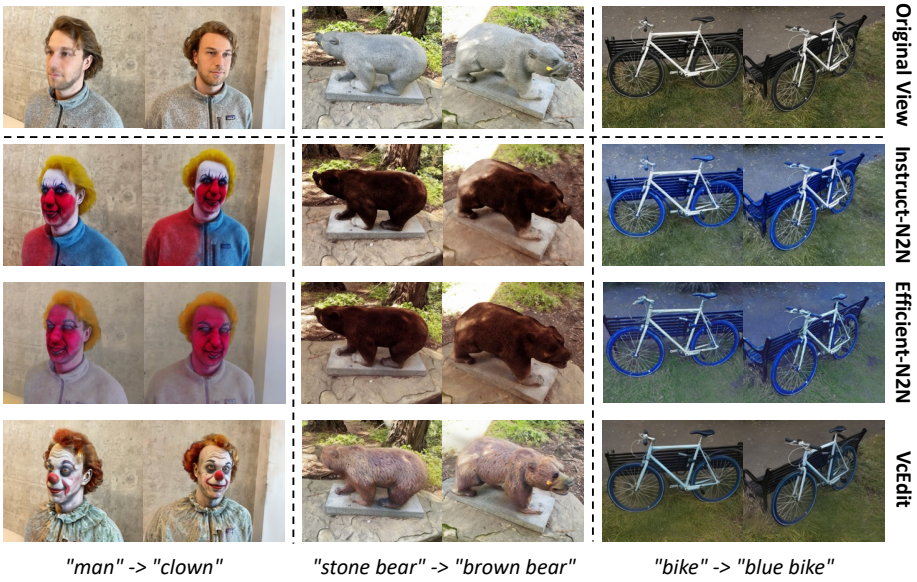

Fig. 3: Qualitative Comparison with the Instruct-NeRF2NeRF [2] and the Efficient-NeRF2NeRF [4]: Our VcEdit outperforms both the NeRF editing methods in terms of achieving finer editing details. Conversely, the NeRF editing approaches produce overly smooth outcomes along with unintended colorization in the background.

3 Qualitative Comparison with NeRF Editing Methods

In our main paper, we compare the editing quality of VcEdit with prevalent 3DGS editing methods, demonstrating that VcEdit significantly surpasses existing state-of-the-art methods in terms of editing quality. To expand our analysis, this section introduces a comparison with NeRF editing approaches, with the findings depicted in Fig. 3. Specifically, we contrast VcEdit with the widely recognized Instruct-NeRF2NeRF [2] and the innovative Efficient-NeRF2NeRF [4].

The comparative analysis depicted in Fig. 3 reveals that NeRF editing methods tend to yield overly smoothed outcomes, in contrast to VcEdit, which delivers results rich in detail. Furthermore, the outcomes of both NeRF editing methods exhibit unintended colorization across the scene, indicating a failure to maintain the original background's integrity. This limitation stems from a fundamental challenge inherent to NeRF models: Compared with 3DGS model, it is more difficult for NeRF to achieve precise local editing. Collectively, these observations underscore VcEdit's superior performance in 3D editing tasks.

4 Detailed Settings of User Study

In our user study, participants encountered a series of questions, each comprising one original view and three corresponding rendered views from the 3DGS edited

via different comparative methods. An illustrative example of such a question is presented in Fig. 4, where participants were tasked with selecting the editing they considered to be of the highest quality. To promote impartiality in responses, the sequence of the methods was randomized for each question, and all options were presented anonymously.

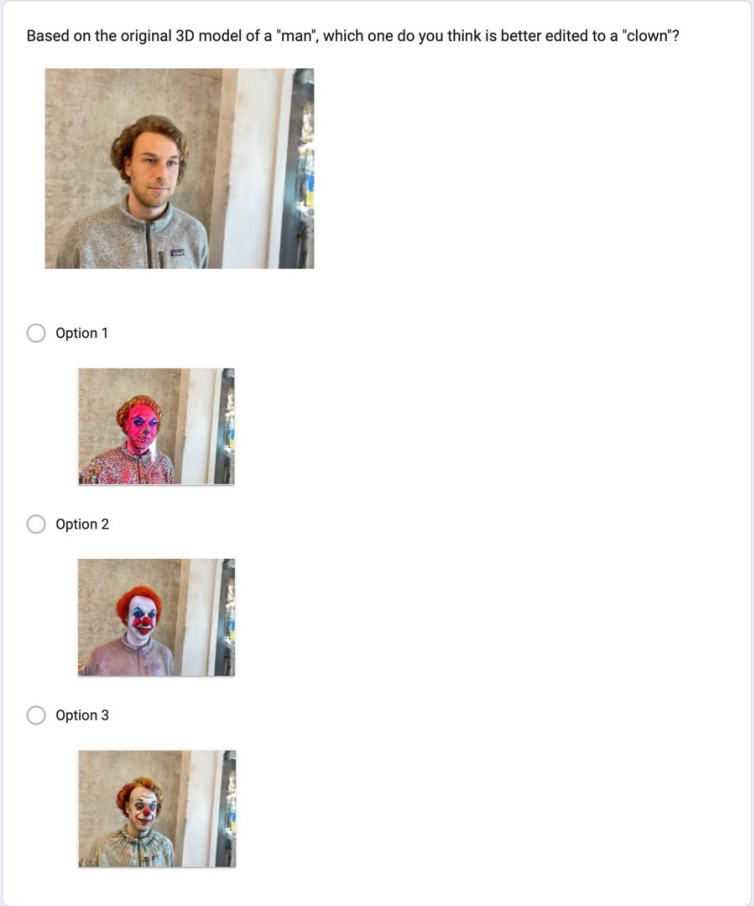

Fig. 4: Question in our User Study: Participants were asked to select one editing they deemed in the best quality.

5 More Editing Results of VcEdit

As the supplementary to our main paper, we present more editing result produced by VcEdit in Fig. 5. Further editing outcomes generated by VcEdit are

displayed in Fig. 5, where our VcEdit produces high-quality editing result for each sample. These results underscore the adaptability of VcEdit in managing a diverse array of complex scenarios and prompts, ranging from detailed facial and object modifications to broad-scale scene alterations. Further video results are released at http://yuxuanw.me/vcedit/.

References

1. Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H., Lin, G.: Gaussianeditor: Swift and controllable 3d editing with gaussian splatting (2023) 2
2. Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions (2023) 5
3. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 2
4. Song, L., Cao, L., Gu, J., Jiang, Y., Yuan, J., Tang, H.: Efficient-nerf2nerf: Streamlining text-driven 3d editing with multiview correspondence-enhanced diffusion models. arXiv preprint arXiv:2312.08563 (2023) 5
5. Xu, S., Huang, Y., Pan, J., Ma, Z., Chai, J.: Inversion-free image editing with natural language (2023) 1, 2, 3

Fig. 5: More Editing Results of VcEdit: Our method is capable of various editing tasks, including face, object, and large-scale scene editing. The *leftmost column* demonstrates the original view, while the *right four columns* show the rendered view of edited 3DGS.