

HDEdit: Supplementary Material

Yanming Zhang^{1,2†}

Jun-Kun Chen^{1†}

Jipeng Lyu¹

Yu-Xiong Wang¹

¹University of Illinois Urbana-Champaign

²University of Maryland

[†]Equal Contribution

{junkun3, jipeng2, yxw}@illinois.edu

yanmingz@umd.edu

immortalco.github.io/HDEdit

This document contains additional analysis and extra experiments.

A Project Page and Supplementary Video

B Video Editing Baselines for 3D Scene Editing

C Implementation Details

- C.1. Settings and Hyperparameters
- C.2. Parallel Denoising Synchronization
- C.3. Attention Map Management
- C.4. Flash Attention-Based Optimization for Attention Map Replacement
- C.5. Attention Control for Long and Looping Videos
- C.6. Case Study: LLM Decomposition

D Additional Results

E Discussion

- E.1. Limitations
- E.2. Overcoming Limitations of Image-Based Editing Methods
- E.3. Future Directions

A. Project Page and Supplementary Video

To better visualize our results and baseline comparisons as videos, we provide a [Project Page](https://immortalco.github.io/HDEdit) on immortalco.github.io/HDEdit.

We also provide a supplementary video as `demo.mp4`, which contains the following three sections:

- Section X: Video Editing Tasks.
- Section Y: 3D Scene Editing Tasks. This section also includes “using video editing baselines to edit 3D scenes.”
- Section Z: Ablation Study.

B. Video Editing Baselines for 3D Scene Editing

The qualitative results of video editing baselines for 3D scene editing are on our [Project Page](https://immortalco.github.io/HDEdit), and the quantitative results are in Tab. A. Our HDEdit significantly outperforms

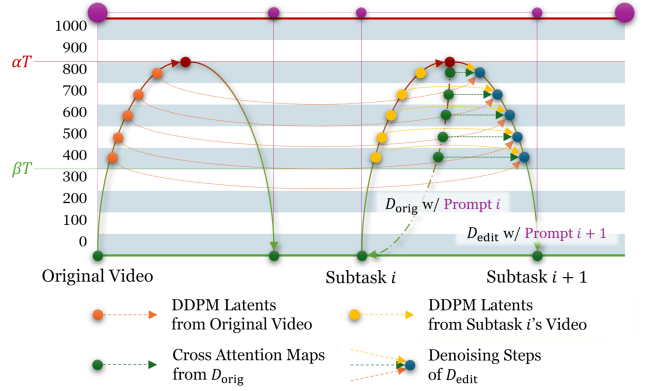


Figure A. When moving forward to the next subtask, our HDEdit performs a parallel denoising control. By reusing the information from previous denoising steps, we effectively preserve the necessary information.

all these methods, especially in the GPT score, which evaluates the overall quality. Also, as shown on our [Project Page](https://immortalco.github.io/HDEdit), even when the videos edited by these baselines have reasonable appearance, the 3D reconstruction is still low-quality and contains lots of artifacts. This shows that our HDEdit is the only model that has sufficient editing capability and preservation control ability to edit 3D scenes.

C. Implementation Details

C.1. Settings and Hyperparameters

GPUs. All of our experiments are run on NVIDIA A6000, A100, and H100 GPUs. Each instance of editing task only needs one GPU.

Underlying Video Diffusion Model. We use CogVideoX-5b [15] as our underlying diffusion model, and fetch the pre-trained weights from HuggingFace. The model is run in pure Brain Float16 (“bfloat16”) data type. We consistently set the classifier-free guidance scale [6] as 7.0, and use CogVideoX’s default CogVideoXDPMSScheduler.

Method	CTIDS \uparrow	CDC \uparrow	GPT Score \uparrow
Edited Video (Intermediate Results of 3D Scene Editing)			
BIVDiff [13]	0.1656	0.0813	51.00
Instruct 4D-to-4D [12]	0.0224	0.1723	29.60
VideoShop [3]	0.0068	-0.0389	21.60
Slicedit [1]	-0.0064	0.1817	31.00
CSD [8]	-0.0035	0.1930	30.60
CogVideoX-V2V [15]	0.2125	0.0554	67.80
HDEdit (Ours)	0.2796	0.1934	90.20
Rendered Edited Scenes			
BIVDiff [13]	0.0901	0.0691	44.00
Instruct 4D-to-4D [12]	-0.0117	0.1507	25.20
VideoShop [3]	0.0045	0.0271	14.20
Slicedit [1]	-0.0061	0.1728	24.00
CSD [8]	0.0035	0.1950	23.60
CogVideoX-V2V [15]	0.2042	0.0266	52.80
HDEdit (Ours)	0.2817	0.2012	90.60

Table A. Quantitative evaluation shows that our HDEdit significantly outperforms all the baselines that use video editing methods for 3D scene editing. This validates that our HDEdit is uniquely capable of performing 3D scene editing among video-based methods.

C.2. Parallel Denoising Synchronization

In our parallel denoising control method, we aim to preserve certain content from the previous video during the denoising generation process. To achieve this, HDEdit employs two control mechanisms: (1) controlling the initial noise to maintain low-frequency information; (2) regulating the noise added at each denoising step to preserve semantic details. These mechanisms, along with our attention map management method (Sec. C.3), work synergistically to maintain the integrity of the original video while enabling effective edits, ensuring successful progression across various editing tasks. A visualization of our parallel denoising control method is shown in Fig. A.

Basic Formulation. We utilize a diffusion model with T noise-adding steps to generate videos in formats such as RGB or latent representations, referred to as “noisy videos” for generality. The noise-adding steps are denoted as A_1, A_2, \dots, A_T , and the denoising steps as D_T, D_{T-1}, \dots, D_1 . Each denoising step D_i involves a denoising network and a noise scheduler. Let v_i represent the video after the i -th noise-adding step, where v_0 is the original video and v_T is pure Gaussian noise. Formally, we define $v_i = A_i(v_{i-1})$ and $v_{i-1} = D_i(v_i)$ for $i = 1, 2, \dots, T$.

Initial Noise Control. To preserve the original video’s overall layout during editing, HDEdit controls the initial noise in the diffusion process. Inspired by SDEdit [11], instead of starting generation from pure Gaussian noise with T denoising steps, we limit noise addition to the first αT steps and then perform denoising from this controlled noise with αT denoising steps. This approach maintains low-frequency information, such as the video’s structure and

layout, while allowing higher-frequency details to be destroyed and regenerated.

Per-Step Noise Control. The method above inspires us that the noise a diffusion model uses also carries semantic information. Building on this observation, HDEdit leverages the noise added at each denoising step to preserve the original video content. Specifically, we utilize DDPM inverse [7] to extract DDPM latents $n_1, n_2, \dots, n_{\alpha T}$ from the original video during the initial αT noise-adding steps, which are the noises to be added at each step in a DDPM denoising procedure. These latents encapsulate rich semantic details essential for maintaining the video’s integrity. By applying them in the corresponding denoising steps $D_{\alpha T}, \dots, D_{\beta T}$, we ensure that semantic information is preserved without excessively constraining high-frequency details, allowing for effective and smooth edits.

However, the DDPM scheduler is inefficient for practical applications. To address this, we explore the intrinsic properties of DDPM inverse, which involves constructing noisy videos and solving for the precise noise required to denoise each step. By defining the denoising function as $D_i(v_i | n_i) \stackrel{\text{Def}}{=} D_i(v_i) + n_i$ for schedulers that do not require random noise n_i , we adapt our preservation control to more advanced and efficient schedulers like DDIM [14] and DPMSolver++ [9, 10]. This novel adaptation allows HDEdit to benefit from the semantic preservation capabilities of the DDPM inverse, while leveraging the high efficiency of advanced denoising schedulers.



Figure B. Our HDEdit produces high-quality editing results in various video editing tasks, achieving superior visual appearance while effectively preserving original content. In contrast, the baselines often introduce visual artifacts, generate unrealistic appearances, or fail to retain regions unrelated to the editing. Notably, CogVideoX-V2V [15], the official video-to-video editing model of CogVideoX, generates visually appealing results but lacks content preservation. This highlights that the strength of HDEdit stems not from the underlying CogVideoX backbone, but from our novel task decomposition and progression framework and preservation control mechanisms. Please refer to [Project Page](#) for the corresponding videos.

C.3. Attention Map Management

Our denoising parallel method achieves preservation by leveraging the clues of visual appearance within the initial noise and DDPM latents, independent of the semantic meaning of the actual subtasks. Our attention map management, in contrast, focuses on another aspect: the overlapping words of the text prompts of the two adjacent high-level subtasks. Following the high-level scheduling, each adjacent pair of high-level subtasks’ prompts exhibits numerous overlaps, allowing us to explicitly consider preservation based on these overlaps.

Cross-Attention Maps for Generation Control. Inspired by attention map replacement strategies of prompt-to-prompt [5], we propose our cross-attention map management. When we work on a subtask moving from state s_{i-1} to s_i , we consider their two adjacent high-level subtasks s_l and s_r , where $l \leq i-1 < i \leq r$. s_{i-1} and s_i are both a linear interpolation of s_l and s_r . Therefore, the overlapping words of s_l and s_r appear in all of these interpolated subtasks. To utilize the overlaps, inspired by [5], we store the cross-attention maps in the generation D_l

of s_l between the overlapping words and the video pixels, and reuse all of them in the generation of both s_i and s_{i-1} . Furthermore, as shown in Fig. ??, the attention map of the overlapping words between s_l and even earlier subtasks can also be stored, maintaining a historical context. Using this strategy, we can preserve not only the data from the previous subtask, but also from a even earlier subtask. In certain cases, such attention map reuse and replacement can be accelerated with Flash Attention (Sec. C.4).

C.4. Flash Attention-Based Optimization for Attention Map Replacement

In dot-product attention

$$\text{Attn}(Q, K, V) = \text{SoftMax}\left(QK^\top/\sqrt{d}\right)V,$$

where Q is $n \times d$, K is $m \times d$, and V is $m \times d'$, the most expensive operation is to explicit construct $M = (QK^\top/\sqrt{d})$, a.k.a. “the attention map”, which is $n \times m$. However, the SoftMax operation does not allow us to directly compute the output. The key insight of flash attention [2] optimization is as follows: consider each row of

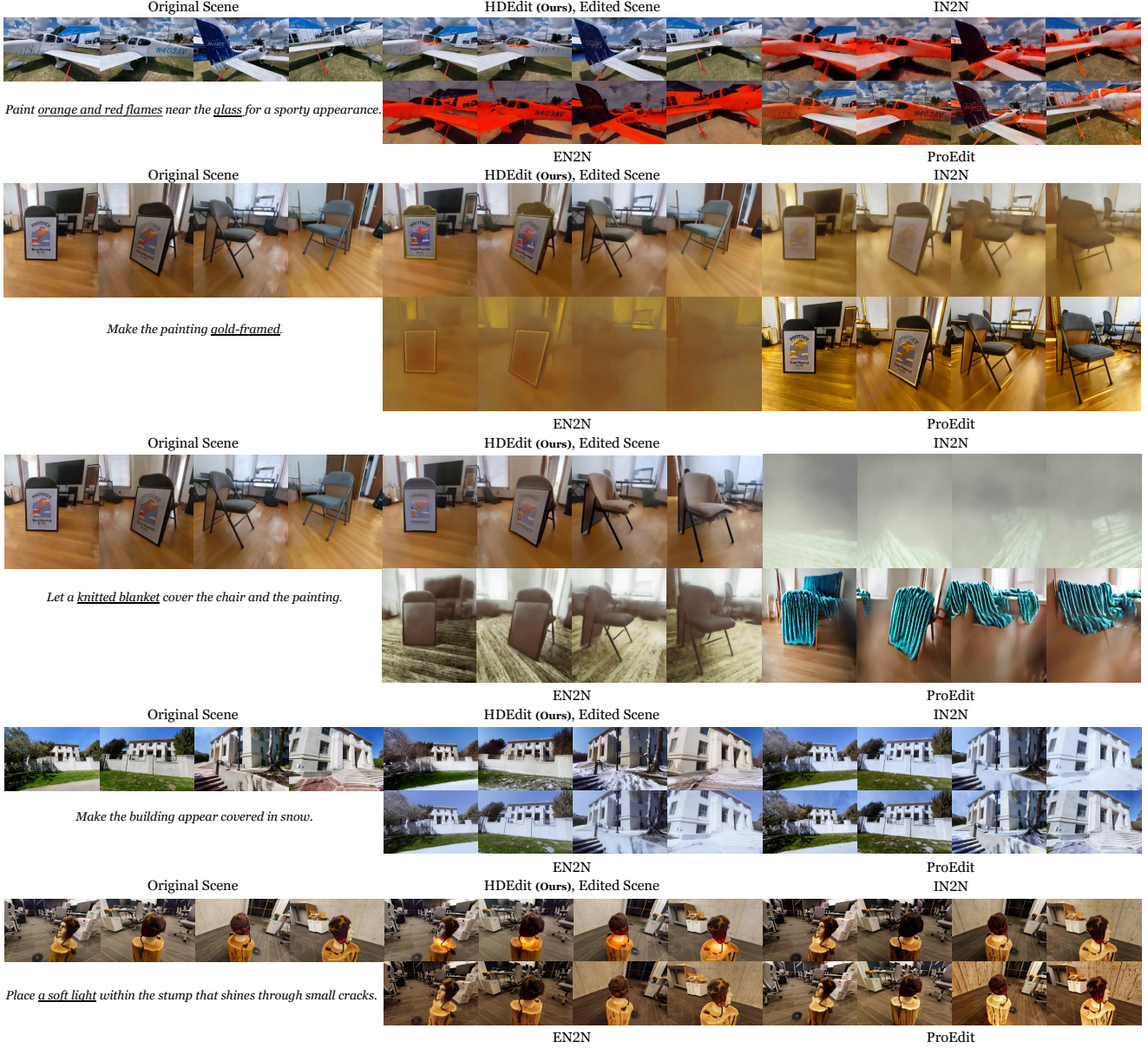


Figure C. Our HDEdit achieves high-quality editing results across various indoor and outdoor scenes, consistently fulfilling editing instructions while preserving original content for all the tasks. In contrast, the baselines either fail to complete the editing or alter many unrelated regions without adequate preservation. Please refer to [Project Page](#) for the rendered videos.

SoftMax(M), we have

$$\text{SoftMax(M)}_{i,j} = \frac{\exp(M_{i,j})}{\sum_j \exp(M_{i,j})}.$$

Therefore, we can first compute the denominator $d_i = \sum_j \exp(M_{i,j})$ for each row i , and then directly calculate $\text{SoftMax(M)}_{i,j} = \exp(M_{i,j})/d_i$. To avoid the precision issue of $\exp(M_{i,j})$ when $M_{i,j}$ is large, we first compute $m_i = \max_j M_{i,j}$, and then define $d'_i = \sum_j \exp(M_{i,j} - m_i)$

to take the place of d_i , so that

$$\text{SoftMax(M)}_{i,j} = \exp(M_{i,j} - m_i)/d'_i.$$

At last, the final output

$$\begin{aligned} \text{Attn}(Q, K, V)_i &= (\text{SoftMax(M)}V)_i \\ &= \sum_j \text{SoftMax(M)}_{i,j} v_j \\ &= \sum_j \exp(M_{i,j} - m_i) v_j / d'_i \end{aligned}$$



Figure D. Ablation study on our progressive framework shows that progression is crucial to obtain high-quality editing results. Notably, the progressive editing process demonstrates a gradual procedure of editing to construct the clock on the wall, ending up with a clock with even 3D-consistent clock hands.

All the elements $M_{i,j}, m_i, d'_i$ can be computed at the same time complexity but without explicitly constructing the whole matrix M , which significantly saves the memory cost by reducing the additional memory complexity from $O(nm)$ to $O(n)$, and the time to allocate and access the memory.

When we use attention map replacement, the only difference is that for the computation of M , some columns (corresponding to K, V , which is the prompt token sequences) need to be replaced from another M' . In this case, we perform the cross-attention of both operations in parallel, and replace the attention map *on-the-fly*. We define

$$\text{AttnAMR}(Q^{(1)}, K^{(1)}, V^{(1)}, Q^{(2)}, K^{(2)}, V^{(2)}, I^{(1)}, I^{(2)})$$

as such function, where $\text{Attn1} = (Q^{(1)}, K^{(1)}, V^{(1)})$ is the attention operation to be computed as usual, and $\text{Attn2} = (Q^{(2)}, K^{(2)}, V^{(2)})$ is the attention operation to be computed with attention-map replacement (AMR), $I^{(1)}, I^{(2)}$ is the index list that the $I_k^{(2)}$ -th column of attention map $M^{(2)}$ of Attn2 should be replaced with the $I_k^{(1)}$ -th column of attention map $M^{(1)}$ of Attn1 for each $1 \leq k \leq |I^{(1)}| = |I^{(2)}|$. In the computation, we only need to simply replace the

computation of $M^{(2)}$ to fit the rule of attention map replacement, *i.e.*, calculate $M_{i,j}^{(2)}$ with $Q_i^{(1)} K_{j'}^{(1)\top}$ instead of $Q_i^{(2)} K_j^{(2)\top}$ if $j = I_k^{(2)}, j' = I_k^{(1)}$ for some k . In this way, we extend the optimization of flash attention to the attention with attention-map replacement.

In our experiments, this optimization could bring a 2 ~ 4 times speed-up.

C.5. Attention Control for Long and Looping Videos

Most of the video diffusion models are trained to generate fixed-length videos. For example, CogVideoX is trained to generate 49-frame videos at 8 FPS. However, some videos to be edited are longer than this length, especially in 3D editing tasks. If we speed up the video to fit the frames, the camera movement will be too fast and, therefore, introduce many challenges for the editing operation to be successful. To address this, we propose a way to enable the support of long video in a training-free manner.

One naive way to generate a long video with a diffusion model is just to extend the size of the input noise, which will not encounter out-of-memory issue with our flash attention-based optimization. However, as the model is only trained on 49-frame videos, it gets confused with the long-term temporal positional embeddings, which are unseen in the training dataset and may lead to low-quality videos. In the generation, this happens in the self-attention, where both Q and K correspond to video patch tokens, and the temporal embedding of Q_i and K_j are too far away. Also, as the K is much larger than the trained situation, the attention also aggregates too much elements from K and further lead to blurred results.

Our insight is that, we can constraint the set of video tokens $\{K_j\}$ that can be seen by each video token Q_i , so that it will only encounter the patterns of positional embeddings that seen in training, and also not see and aggregate too many K_j to make the video blurred. More specifically, if the Q_i belongs to frame k , we allow Q_i to see all the tokens within frames $[k - l/2, k + l/2]$, where l is the length of the pre-trained videos. With this control, each Q_i will only see the tokens within nearby l frames, which are seen patterns in training, and also not aggregating too much $\{K_j\}$ s – which will be no more than l frames, *i.e.* the number of $\{K_j\}$ s seen in training. With this optimization, the model is able to generate high-quality long videos in a training-free manner.

In 3D scene editing, some of the camera trajectories are looping, *i.e.*, the first frame continues the last frame. In this case, it is desirable to also make the edited video looping. Therefore, when generating the first several frames, *e.g.* frame i where $i < l/2$, we allow it to see the last several frames $L + i - l/2 < j \leq L$, where L is the total length of the current video, and apply the temporal posi-

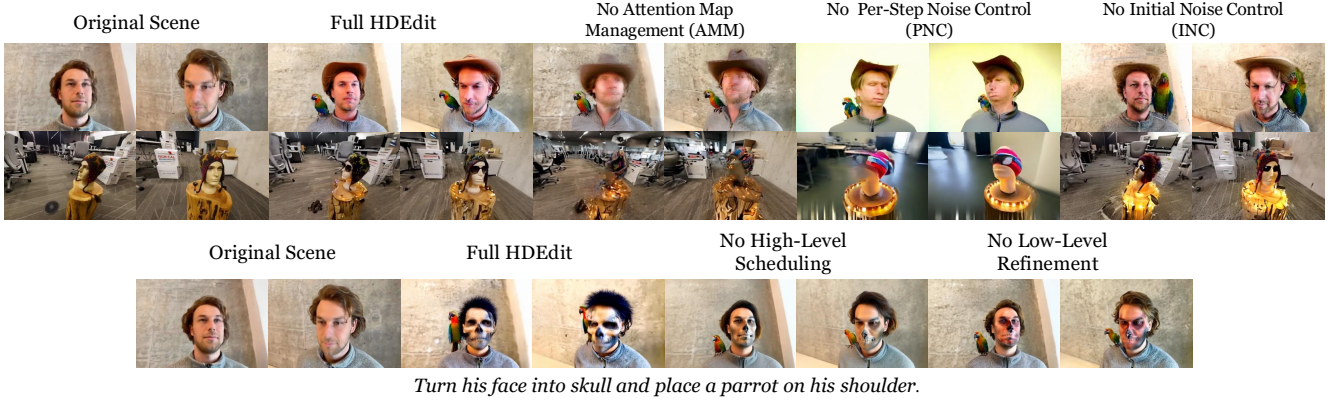


Figure E. Ablation studies show that all our strategies are crucial to successful and high-quality editing results. Removing any of them results in significant degradation. The video visualizations are in our [Project Page](#).

tional embedding of $j - L$ to frame j , to make it look like one of the previous frames of i to the model. In this case, the model learns to generate looping edited videos at few additional computation cost, and significantly improves the consistency.

C.6. Case Study: LLM Decomposition

We provide a case study to visualize the high-level decomposition process. The editing task is the “mustache, parrot, adventurer” task in Fig. This task is decomposed into three subtasks:

- **State 0 (Initial).**
“A person with light golden brown hair; a normal face, and a light beard, dressed in a gray zip-up jacket. He stands calmly against a minimalist concrete wall. The camera starts from a fixed position, then slowly rotates around the person, capturing their profile and facial expressions as the perspective shifts smoothly, with the plain, textured background highlighting the subject.”
- **State 1 (Medium, 2 low-level subtasks).**
*“A person with light golden brown hair, **sporting a thick, big mustache that curls at the ends**, dressed in a gray zip-up jacket. He stands calmly against a minimalist concrete wall. The camera starts from a fixed position, then slowly rotates around the person, capturing their profile and facial expressions as the perspective shifts smoothly, with the plain, textured background highlighting the subject.”*
- **State 2 (Final; Hard, 3 low-level subtasks).**
*“A person with light golden brown hair, **sporting a thick, big mustache that curls at the ends**, dressed in a gray zip-up jacket, **with a colorful parrot statue perched confidently on his shoulder, enhancing his adventurous look**. He stands calmly against a minimalist concrete wall. The camera starts from a fixed position, then slowly rotates around the person, capturing their profile and facial*

expressions as the perspective shifts smoothly, with the plain, textured background highlighting the subject.”

This example shows how HDEdit assigns sensible difficulty scores and progressively fulfills complex edits.

D. Additional Results

We present additional experimental results.

- Tab. **B**: Ablation study results.
- Fig. **B**: Additional video editing results.
- Fig. **C**: Additional 3D scene editing results.
- Figs. **E,D**: Additional ablation study results.

E. Discussion

E.1. Limitations

The limitations of our HDEdit primarily lie in two aspects: the editing capability limited by the underlying video diffusion model, and the 3D awareness and motion constraints in 3D editing. It is important to note that most image- and video-based editing methods also encounter these challenges.

Editing Capability. As a general framework, our HDEdit is compatible with various underlying video diffusion models. When paired with a specific model, *e.g.*, CogVideoX [15], the editing capability of our framework inherits the strengths and limitations of that model. For example, as CogVideoX is trained exclusively on 720×480 landscape videos, our HDEdit using CogVideoX supports only landscape videos, not portrait videos. Also, if a diffusion model cannot recognize a specific concept, such as an object or a type of motion, we cannot perform effective editing related to that concept. Similarly, if the underlying model fails to generate a plausible video for a given prompt, our preservation control cannot produce a reasonable edited video based

Method	CTIDS \uparrow	CDC \uparrow	GPT Score \uparrow
Video Editing			
w/o Decomposition & Progression (Pro)	0.0611	0.2414	65.23
w/o Parallel Denoising Control – Initial Noise Control (INC)	0.0763	0.1922	64.80
w/o Parallel Denoising Control – Per-Step Noise Control (PNC)	0.0949	0.0291	55.83
w/o Attention-Map Management (AMM)	0.0898	0.1079	51.04
Full HDEdit (Ours)	0.1029	0.2933	76.60
3D Scene Editing			
w/o Decomposition & Progression (Pro)	0.0571	0.2744	59.17
w/o Parallel Denoising Control – Initial Noise Control (INC)	0.0655	0.2462	52.50
w/o Parallel Denoising Control – Per-Step Noise Control (PNC)	0.0789	0.0088	34.17
w/o Attention-Map Replacement (AMM)	0.0829	0.1109	53.33
Full HDEdit (Ours)	0.0954	0.3658	77.50

Table B. Ablation study shows that all our strategies are crucial to our final performance.

on the corresponding editing instruction. Adopting a more advanced video diffusion model could help mitigate this limitation. Additionally, our interpolation-based decomposition offers a potential pathway to bypass unsupported concepts by breaking down the task into simpler, manageable subtasks.

3D Editing: 3D Awareness and Motion Constraint. In 3D editing, the edited video should ideally represent a rendered video of a 3D scene, maintaining 3D consistency and remaining static. However, since the video diffusion model lacks 3D input, it is not 3D aware. As a result, the edited video may not be 3D-consistent. On the other hand, the edited video may contain some motion, which is not desired for 3D scene editing. Both limitations can be mitigated by our progression-based generation procedure. At each subtask, by reconstructing the edited video into the 3D scene and then re-rendering the video, we enforce the 3D consistency and ensure the video remains static at this step. By decomposing the process into subtasks with minimal modifications, such inconsistency and unwanted motion are reduced to minor levels, which can be corrected during reconstruction and re-rendering.

E.2. Overcoming Limitations of Image-Based Editing Methods

Despite the limitations mentioned above, our video-based HDEdit framework successfully overcomes the limitations of image-based (per-frame/per-view) editing methods. While the image-based methods require numerous iterations (as seen in the “iterative dataset update” in [4]) to achieve convergence, our HDEdit can directly produce edited videos based on an end-to-end framework without time-consuming iterations. In 3D scene editing, our HDEdit leverages video diffusion models to its advantage. First, the generated video is smooth and temporally consistent, which is a strong prerequisite for achieving 3D consistency. This

allows us to directly reconstruct the video into 3D, eliminating the need for iterative processes and enabling large-scale shape editing. Additionally, by analyzing the entire video, which contains a complete view of scene objects, our HDEdit avoids common issues like Janus or multi-face artifacts that plague image-based editing methods. It also supports view-dependent effects, such as specular effects generated by the video diffusion model, and successfully reconstructs them in 3D. By overcoming these challenges, our HDEdit produces state-of-the-art results in both video and 3D scene editing.

E.3. Future Directions

4D Scene Editing. While a video represents “dynamic 2D” and a 3D scene represents “static 3D,” one interesting direction is to extend this work to 4D, encompassing “dynamic 3D.” This would involve complex editing tasks such as motion editing, object moving, *etc.*

3D-Aware Video-Based Scene Editing. Another direction is to make the video diffusion model 3D-aware for 3D scene editing. This could be achieved by training a new video diffusion model on RGBD videos and integrating it with our HDEdit. Doing so would enable 3D awareness in the model and potentially make it easier to control the video content stationary as a rendered 3D video.

References

- [1] Nathaniel Cohen, Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Slicedit: Zero-shot video editing with text-to-image diffusion models using spatio-temporal slices. *arXiv:2405.12211*, 2024. 2
- [2] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. *arXiv:2205.14135*, 2022. 3
- [3] Xiang Fan, Anand Bhattad, and Ranjay Krishna. Videoshop:

Localized semantic video editing with noise-extrapolated diffusion inversion. *arXiv:2403.14617*, 2024. 2

- [4] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 7
- [5] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv:2208.01626*, 2022. 3
- [6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1
- [7] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly DDPM noise space: Inversion and manipulations. In *CVPR*, 2024. 2
- [8] Subin Kim, Kyungmin Lee, June Suk Choi, Jongheon Jeong, Kihyuk Sohn², and Jinwoo Shin¹. Collaborative score distillation for consistent visual editing. In *NeurIPS*, 2023. 2
- [9] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *arXiv:2206.00927*, 2022. 2
- [10] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv:2211.01095*, 2023. 2
- [11] 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
- [12] Linzhan Mou, Jun-Kun Chen, and Yu-Xiong Wang. Instruct 4D-to-4D: Editing 4D scenes as pseudo-3D scenes using 2D diffusion. In *CVPR*, 2024. 2
- [13] Fengyuan Shi, Jiaxi Gu, Hang Xu, Songcen Xu, Wei Zhang, and Limin Wang. BIVDiff: A training-free framework for general-purpose video synthesis via bridging image and video diffusion models. *arXiv:2312.02813*, 2024. 2
- [14] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, 2020. 2
- [15] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. *arXiv:2408.06072*, 2024. 1, 2, 3, 6