

---

## APPENDIX

### A EXPERIMENTAL DETAILS

We use an NVIDIA RTX 6000 Ada graphics card for all training and inference tasks.

#### A.1 MULTI-TO-MULTI

In the Multi-to-Multi setting, we utilize the DDIM Inversion scheduler (Song et al., 2020) with 50 inversion steps for all methods. We use Stable Diffusion v1-5<sup>1</sup> as the pre-trained multi-step model.

**FLATTEN** (Cong et al., 2024). We use the official PyTorch implementation of FLATTEN<sup>2</sup>, following the settings from the original paper. We set the guidance scale to 20 and use 50 sampling steps.

**TokenFlow** (Geyer et al., 2023). We use the official PyTorch implementation of TokenFlow<sup>3</sup>. Following the paper’s methodology, we utilize Plug-and-Play editing (Tumanyan et al., 2023) and set the guidance scale to 7.5 with 50 sampling steps.

**FRESCO** (Yang et al., 2024). We use the official PyTorch implementation of FRESCO<sup>4</sup>. As recommended in the paper, we use ControlNet with HED conditioning<sup>5</sup>, 20 sampling steps, and a guidance scale of 7.5. We select a keyframe every 8 frames and use EbSynth<sup>6</sup> for propagation and final editing.

**RAVE** (Kara et al., 2024). We use the official PyTorch implementation of RAVE<sup>7</sup>. Following the paper, we employ a ControlNet conditioned on depth maps<sup>8</sup> with a grid size of 3, a guidance scale of 7.5, and 50 sampling steps.

**COVE** (Wang et al., 2024). We use the official PyTorch implementation of COVE<sup>9</sup>. Following the settings from the paper, we set the guidance scale to 7.5, the DIFT up-sampling index to 7.5, the window size to 7, the correspondence-guidance scale to 3, and the token merging ratio to 50%.

#### A.2 MULTI-TO-ONE

In the Multi-to-One setting, we use DMD2 (Yin et al., 2024) as our pre-trained one-step model. For all methods, we use the DDIM Inversion scheduler (Song et al., 2020) with 50 inversion steps. Inference is performed in a single step for all experiments in this setting.

**Prompt Replacement.** This baseline method involves replacing the source prompt (e.g., “A photo of a dog”) with the target prompt (e.g., “A photo of a cat”).

**Prompt-to-Prompt** (Hertz et al., 2023). We use the official PyTorch implementation of Prompt-to-Prompt<sup>10</sup>. For our evaluation, we only replace the cross-attention maps, as we found that replacing self-attention maps significantly degraded image quality.

**ControlNet (Depth)** (Zhang & Agrawala, 2023). We use a pre-trained ControlNet model for depth conditioning from Hugging Face<sup>11</sup>. We set the ControlNet injection parameter to 0.8.

**ControlNet (Canny)** (Zhang & Agrawala, 2023). We use a pre-trained ControlNet model for Canny edge conditioning from Hugging Face<sup>12</sup>. We set the ControlNet injection parameter to 0.8.

---

<sup>1</sup><https://huggingface.co/runwayml/stable-diffusion-v1-5>

<sup>2</sup><https://github.com/yrcong/flatten>

<sup>3</sup><https://github.com/omerbt/TokenFlow>

<sup>4</sup><https://github.com/williamyang1991/FRESCO>

<sup>5</sup><https://huggingface.co/lllyasviel/sd-controlnet-hed>

<sup>6</sup><https://github.com/jamriska/ebsynth>

<sup>7</sup><https://github.com/RehgLab/RAVE>

<sup>8</sup><https://huggingface.co/lllyasviel/sd-controlnet-depth>

<sup>9</sup><https://github.com/wangjiangshan0725/COVE>

<sup>10</sup><https://github.com/google/prompt-to-prompt>

<sup>11</sup><https://huggingface.co/lllyasviel/sd-controlnet-depth>

<sup>12</sup><https://huggingface.co/lllyasviel/sd-controlnet-canny>

**Plug-and-Play** (Tumanyan et al., 2023). We use the official PyTorch implementation of Plug-and-Play<sup>13</sup>. We inject the spatial features from the convolutional layers of the U-Net, as described in the paper.

### A.3 ONE-TO-ONE

In the One-to-One setting, both inversion and sampling are performed in a single step using one-step models.

**VEDIS (Ours)**. We use our trained VEDIS encoder for one-step inversion, passing the predicted noise directly to the DMD2 (Yin et al., 2024) one-step diffusion model. In this setting, edits are performed using only the target text prompt, as our inversion encoder is designed to preserve the structure of the source image.

Table 1: Ablation on noise injection parameter  $\lambda_{\text{noise}}$ .

$\lambda_{\text{noise}}$	Structure Distance ( $\downarrow$ )	CLIP Whole ( $\uparrow$ )	CLIP Edit ( $\uparrow$ )
0.0	0.087	21.797	19.884
0.1	<b>0.064</b>	22.329	20.416
0.2	0.068	22.314	20.406
0.3	0.075	22.449	20.422
0.4	0.085	22.531	20.402
0.5	0.101	22.641	20.424

## B ABLATION ON PROMPT PERTURBATION

Table 1 presents an ablation study on the noise injection parameter for the SAE loss. The results indicate that a parameter of 0.1 achieves the lowest structure distance, signifying the best preservation of the source content’s geometry. Conversely, as the injection parameter increases, the CLIP score also rises. This suggests a trade-off: a larger parameter pushes the inverted latent to be more editable and better aligned with the target prompt, but at the cost of structural fidelity. Given that our primary objective is to perform edits while faithfully preserving the original structure, we set this parameter to 0.1 for our main experiments.

## C ENCODER TRAINING

We trained the inversion encoder for 72 hours using a batch size of 6. The weight for the SAE loss,  $\lambda_{\text{sae}}$ , was set to 1. For optimization, we used the AdamW (Kingma, 2014) optimizer with a learning rate of 1e-05, betas of (0.9, 0.999), an epsilon of 1e-8, and a weight decay of 1e-2. The loss curves during training are visualized in Figure 1 and 2.

Table 2: Ablation of Anchor methods (90 Frames).

Anchor method	90 Frames					
	SC	BC	MS	AQ	IQ	BQS
Random	0.950	0.963	0.987	0.670	0.723	0.673
First	0.958	0.962	0.988	0.670	0.723	0.675
Pixel-Centroid	0.957	0.962	0.988	0.670	0.723	0.675
Pixel-Medoid	0.957	0.964	0.988	0.670	0.723	0.675
<b>DINO-Medoid</b>	0.958	0.965	0.989	0.670	0.723	0.676
CLIP Top-1	0.957	0.963	0.988	0.670	0.723	0.675
Hybrid	0.956	0.962	0.988	0.670	0.723	0.675

<sup>13</sup><https://github.com/MichalGeyer/plugin-and-play>

## D ANCHOR ABLATION

To determine the optimal anchor selection strategy, we conducted an ablation study comparing our chosen method against seven alternatives. The evaluated methods include: 1) *Random*, which selects a random frame; 2) *First*, which always uses the initial frame; 3) *Pixel-Centroid*, which selects the frame closest to the pixel-wise mean of all frames; 4) *Pixel-Medoid*, which selects the frame minimizing the sum of pixel-wise distances to all other frames; 5) *DINO-Medoid*, which identifies the medoid in the DINO (Caron et al., 2021) feature space; 6) *CLIP Top-1*, which selects the frame with the highest CLIP (Radford et al., 2021) similarity to the source prompt; and 7) *Hybrid*, which considers both DINO and CLIP scores. The ablation was performed on 90-frame videos to capture the long-term effects of the anchor choice. As shown in Table 2, the DINO-Medoid strategy consistently yielded the best performance, achieving the highest scores across our metrics.

## E ABLATION ON SLIDING WINDOW AND ANCHOR

In this section, we conduct an ablation study on the sliding window size and stride to find the optimal balance between editing quality and processing speed. As shown in Table 3, a window size of 7 and a stride of 5 (with an anchor) provides a strong trade-off, which we adopt for our main experiments.

Table 3: Ablation on Unified-Frame-Editing on 90 Frame videos.

Window size $w$	Stride	w/o Anchor								w/ Anchor							
		SC	BC	MS	AQ	IQ	BQS	FPS		SC	BC	MS	AQ	IQ	BQS	FPS	
1	1	0.931	0.948	0.980	0.680	0.723	0.668	20.925	0.943	0.950	0.985	0.680	0.722	0.669	15.332		
3	1	0.950	0.962	0.988	0.676	0.723	0.676	11.646	0.958	0.966	0.988	0.010	0.723	0.679	8.698		
5	1	0.952	0.963	0.988	0.674	0.722	0.676	7.068	0.957	0.965	0.988	0.010	0.723	0.678	5.644		
	3	0.952	0.962	0.987	0.673	0.722	0.675	17.409	0.958	0.963	0.988	0.010	0.723	0.677	14.324		
7	1	0.954	0.964	0.988	0.670	0.722	0.674	4.769	0.957	0.966	0.988	0.010	0.723	0.677	3.959		
	3	0.954	0.963	0.988	0.671	0.722	0.674	12.354	0.957	0.964	0.988	0.010	0.723	0.676	10.551		
	5	0.954	0.963	0.988	0.672	0.723	0.675	18.140	0.958	0.962	0.988	0.010	0.723	0.676	15.793		
9	1	0.955	0.963	0.989	0.669	0.723	0.674	3.351	0.957	0.963	0.989	0.010	0.723	0.675	2.870		
	3	0.954	0.962	0.988	0.670	0.723	0.674	9.203	0.957	0.961	0.988	0.010	0.723	0.675	7.886		
	5	0.954	0.965	0.988	0.669	0.723	0.674	13.862	0.957	0.964	0.988	0.010	0.723	0.676	12.109		
	7	0.955	0.964	0.988	0.670	0.723	0.674	17.599	0.958	0.964	0.988	0.010	0.723	0.676	15.614		

## F UNIFIED-FRAME EDITING

We prepare the sequence for sliding window processing. To ensure sufficient context for boundary frames, we apply reflect padding to the sequence of inverted latents  $\hat{\mathcal{V}}_K^T = (\hat{z}_0^T, \dots, \hat{z}_{K-1}^T)$ . Let  $p = (w - 1)/2$  be the padding size (for an odd  $w$ ). The padded sequence  $\hat{\mathcal{V}}_{\text{pad}}^T$  is constructed as:

$$\hat{\mathcal{V}}_{\text{pad}}^T = (\underbrace{\hat{z}_p^T, \dots, \hat{z}_1^T}_{\text{left reflection}}, \underbrace{\hat{z}_0^T, \dots, \hat{z}_{K-1}^T}_{\text{original}}, \underbrace{\hat{z}_{K-2}^T, \dots, \hat{z}_{K-1-p}^T}_{\text{right reflection}})$$

The padded sequence is processed to generate a series of output segments. The total number of segments,  $N_w$ , is given by  $N_w = \lceil K/s \rceil$ . For each segment index  $i \in \{0, 1, \dots, N_w - 1\}$ , we first extract a window of  $w$  latents from the padded sequence. These are spatially concatenated to form the window map  $W_i^T$ :

$$W_i^T = \text{Concat}(\hat{z}_{i \cdot s}, \dots, \hat{z}_{i \cdot s + w - 1}, \text{axis} = 2)$$

The final input for the generator,  $W_i'^T$ , is then constructed by prepending the anchor latent  $\hat{z}_A^T$  to this window map:

$$W_i'^T = \text{Concat}(\hat{z}_A^T, W_i^T, \text{axis} = 2)$$

The generator processes this input, producing a unified output map  $W_i'^0 = G_\theta(W_i'^T, c_{\text{edit}})$ . From this output, we first discard the initial portion corresponding to the anchor frame to get  $W_i^0$ :

$$W_i^0 = W_i'^0[:, :, W:] \quad (\text{Dimensions: } \mathbb{R}^{C \times H \times (W \cdot w)})$$

---

Next, we extract the central  $s$  frames from  $W_i^0$  to form the final segment  $S_i$ :

$$S_i = W_i^0[:, :, W \cdot \text{offset} : W \cdot (\text{offset} + s)] \quad (\text{where } \text{offset} = (w - s)/2)$$

After generating all  $N_w$  segments, they are concatenated. The resulting map is truncated to the original length of  $K$  frames to form the final edited latent map,  $Z_{\text{edit}}^0$ :

$$Z_{\text{edit}}^0 = \text{Truncate}(\text{Concat}(S_0, S_1, \dots, S_{N_w-1}, \text{axis} = 2), K).$$

Finally, this unified map is partitioned back into a sequence of individual frame latents:

$$\hat{z}_{k_{\text{edit}}}^0 = Z_{\text{edit}}^0[:, :, k \cdot W : (k + 1) \cdot W] \quad \text{for } k = 0, \dots, K - 1.$$

The final edited video,  $\mathcal{V}_{\text{edit}}^0$ , is the sequence of these  $K$  latents:

$$\hat{\mathcal{V}}_{\text{edit}}^0 = (\hat{z}_{0_{\text{edit}}}^0, \hat{z}_{1_{\text{edit}}}^0, \dots, \hat{z}_{K-1_{\text{edit}}}^0).$$

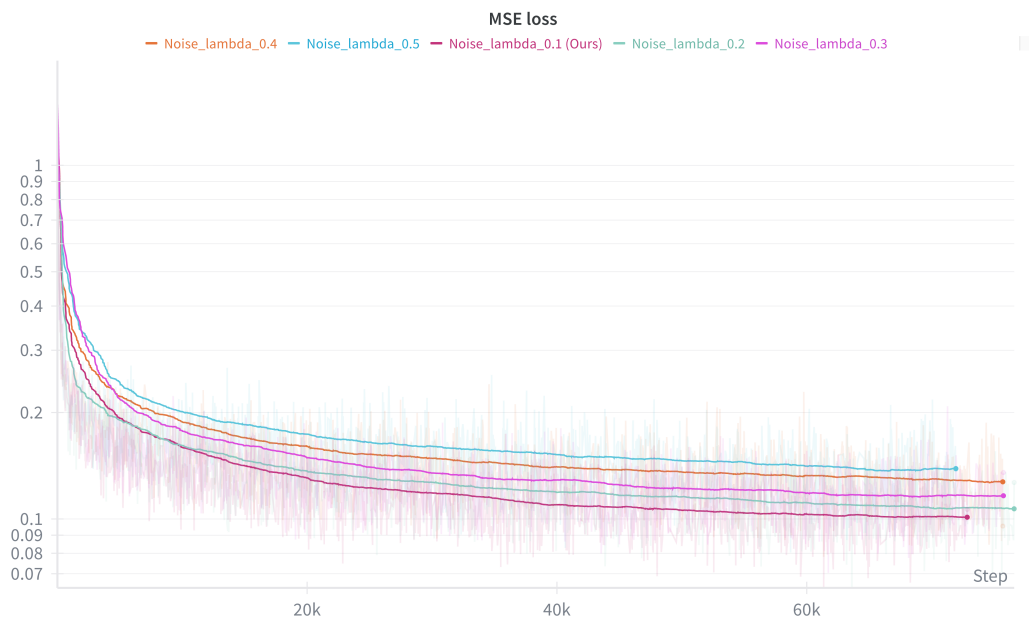


Figure 1: Graph of MSE loss while training.

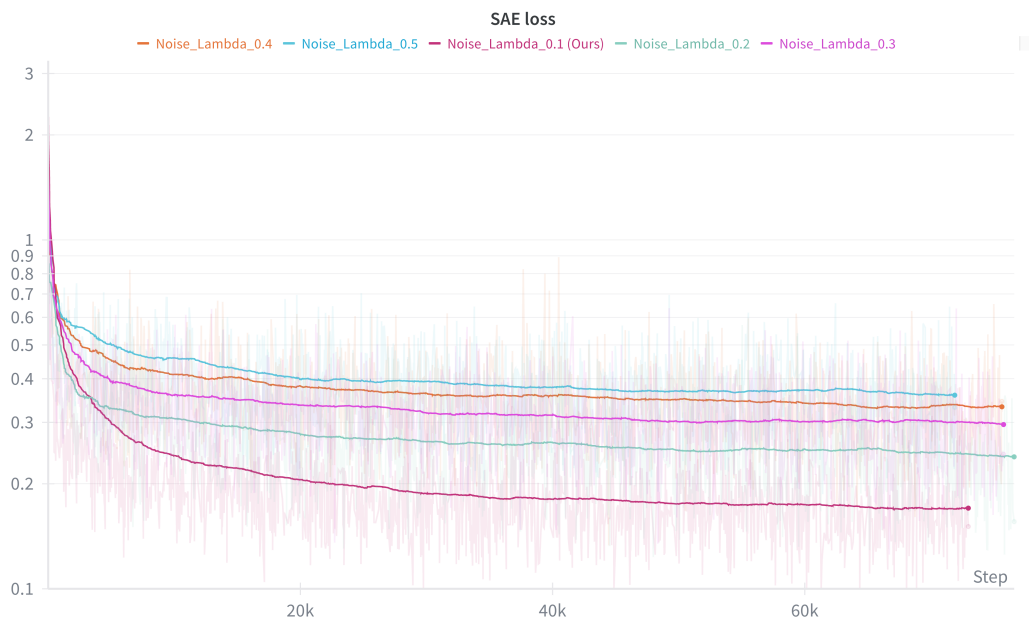


Figure 2: Graph of SAE loss while training.

---

## REFERENCES

- Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.
- Yuren Cong, Mengmeng Xu, Christian Simon, Shoufa Chen, Jiawei Ren, Yanping Xie, Juan-Manuel Perez-Rua, Bodo Rosenhahn, Tao Xiang, and Sen He. Flatten: optical flow-guided attention for consistent text-to-video editing, 2024. URL <https://arxiv.org/abs/2310.05922>.
- Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arXiv:2307.10373*, 2023.
- Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023. URL [https://openreview.net/forum?id=\\_CDixzkzeyb](https://openreview.net/forum?id=_CDixzkzeyb).
- Ozgur Kara, Bariscan Kurtkaya, Hidir Yesiltepe, James M Rehg, and Pinar Yanardag. Rave: Randomized noise shuffling for fast and consistent video editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6507–6516, 2024.
- Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- 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 *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.
- Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1921–1930, 2023.
- Jiangshan Wang, Yue Ma, Jiayi Guo, Yicheng Xiao, Gao Huang, and Xiu Li. Cove: Unleashing the diffusion feature correspondence for consistent video editing. *Advances in Neural Information Processing Systems*, 37:96541–96565, 2024.
- Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Fresco: Spatial-temporal correspondence for zero-shot video translation, 2024. URL <https://arxiv.org/abs/2403.12962>.
- Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024.
- Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.