PropFly: Learning to Propagate via On-the-Fly Supervision from Pre-trained Video Diffusion Models

Supplementary Material

A. Introduction

In this supplementary material, we provide additional details omitted from the main paper. Sec. B and Sec. C elaborate on the implementation details and evaluation protocols, respectively. Sec. D presents in-depth ablation studies, including analyses on CFG scaling and data pair quality, followed by a discussion of limitations in Sec. E. Finally, Sec. F provides extensive qualitative comparisons on the DAVIS [41] and EditVerseBench [23] datasets to further demonstrate the capabilities of PropFly.

B. Implementation Details

B.1. Training Details

We train our PropFly models for a total of 50,000 iterations at a fixed resolution of 480×832 with 33 frames. Optimization is performed using the AdamW optimizer [35] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.1. The learning rate is held constant at 1×10^{-5}. The training is conducted on 4 NVIDIA A100 (80GB) GPUs. For PropFly-14B, the training process takes approximately 12 days with a global batch size of 4. For PropFly-1.3B, it requires approximately 2.5 days with a global batch size of 48. We utilize Bfloat16 precision for both PropFly-14B and PropFly-1.3B to optimize memory usage and training speed.

The pre-trained Wan2.1 backbone [50] remains frozen. We only finetune the following parameters within the DiT adapter blocks: patch embedding parameters, linear projection layers within the attention blocks, including `to_q`, `to_k`, `to_v`, and `to_out`.

B.2. Inference Details

For all inference results, we use a single Classifier-Free Guidance (CFG) scale of $\omega = 1.0$. All source videos were pre-processed by resizing and center-cropping them to a 480×832 resolution. We perform denoising using the UniPC scheduler [62] with 25 sampling steps.

B.3. Computational Complexity

The primary overhead of our PropFly training pipeline stems from the added sampling process required for on-the-fly data pair generation. Since this sampling is fully decoupled from the loss calculation, no gradient backpropagation is required for this step. Consequently, the necessary GPU memory footprint remains unchanged compared to the baseline VDM training setup. The required training time of one iteration for our PropFly-1.3B is 4.95 seconds, compared to the training time with paired datasets of 4.71 seconds. This represents an overhead of approximately 5.1% per iteration. Note that training with paired datasets requires encoding both source and edited videos, while PropFly only needs to encode the original video, thereby reducing the training time gap. As PropFly is a training pipeline designed only to train a lightweight adapter, the computational complexity for inference remains the same as the underlying VDM architecture with the adapter.

B.4. Style Prompt Set

To implement the Random Style Prompt Fusion (RSPF) introduced in Sec. 4.2, we curated a diverse collection of style descriptors to enrich our on-the-fly supervision. We generated an initial candidate list using the Gemini 2.5 Flash model [10] and manually filtered the results to ensure distinctiveness and visual impact. The final set comprises 113 phrases spanning a wide range of categories, including weather, artistic styles, materials, mood, and backgrounds. The complete list is provided in Table 4.

C. Evaluation Details

C.1. Edited First Frame Generation

For edited first frame synthesis, we provide the Gemini 2.5 Flash Image model [10] with the original first frame and the target text prompt. For fair comparison of our PropFly against other propagation-based video editing methods, AnyV2V [30] and Señorita-2M [63], we use the same edited first frames. Since CCEdit [12] propagates the edits from the center frame, we utilized the edited center frame from the EditVerse results. Additionally, for the 'Propagation' category within EditVerseBench, we employ the officially provided edited first frames to align with standard evaluation protocols.

C.2. EditVerseBench

For quantitative comparison, we evaluate our PropFly on the EditVerseBench [23], a recent benchmark for instruction-guided video editing. The full benchmark comprises 100 videos (50 horizontal and 50 vertical) spanning 20 distinct editing categories, resulting in 200 total video-instruction pairs. As our method focuses on visual appearance and style propagation, we utilize a subset of the full benchmark relevant to our setting, referred to as EditVerseBench-Appearance in the main paper. We selected 11 categories relevant to our scope

Table 4. Style prompt set used for RSPF during the training of PropFly.

Weather	Artistic Style	Material	Mood	Backgrounds
snowy	in the style of Van Gogh	made of glass	monochrome	Gradient Background
rainy	in the style of Monet	made of crystal	vibrant colors	Moving Particle Background
drizzling	in the style of Picasso	made of ice	pastel colors	Geometric Pattern Background
foggy	in the style of Dali	glowing	muted colors	Neon Line Background
misty	in the style of Rembrandt	holographic	dreamy	Abstract Digital Art Background
sunny	in the style of Hokusai	iridescent	ethereal	Matrix Code Background
overcast	in the style of Ukiyo-e	metallic	eerie	Plexus Background
stormy	Impressionism	chrome	serene	Watercolor Splash Background
thunderstorm	Surrealism	gold	nostalgic	Smoke or Fog Effect Background
partly cloudy	Cubism	silver	dramatic lighting	Bokeh Effect Background
windy	Pop Art	bronze	soft lighting	Cloudy Sky Time-lapse
hazy	Art Nouveau	wooden	cinematic	Starry Night Sky
golden hour	watercolor painting	marble	epic	Calm Ocean Waves
blue hour	oil painting	stone	minimalist	Rainy Day Window View
night	acrylic painting	fabric	maximalist	Snowfall Scene
dawn	sketch	velvet	dark and moody	Sunbeams in the Forest
dusk	charcoal drawing	clay	light and airy	Aerial Drone View
sunset	pen and ink drawing	origami		Sunset Scenery
sunrise	digital art			Tranquil Lake Scene
midday sun	concept art			Four Seasons Landscape
aurora	pixel art			Paper Texture Background
	vector art			Old Film Effect Background
	low poly			Wood Grain Background
				Fabric Texture Background
				Concrete Wall Background
				Water Drops on Glass Background
				Glitter and Sparkle Effect Background
				City Night Time-lapse
				Library or Study Background
				Cozy Cafe Background
				High-Tech Circuit Board Background
				Newsroom Style Background
				Stage Light and Spotlight Background
				Clean White/Black Studio Background

and excluded tasks unrelated to appearance modification (e.g., 'Change camera pose', 'Detection', 'Pose-to-video', 'Depth-to-video', 'Edit with mask'). The 11 selected categories are: 'Add object', 'Remove object', 'Change object', 'Stylization', 'Propagation', 'Change background', 'Change color', 'Change material', 'Add effect', 'Change weather', 'Combined tasks'.

For assessing video quality, we calculate the PickScore [28] using the CLIP ViT-H/14 [44] backbone. For text-frame alignment, we average the cosine similarities calculated across all frames and the text instruction, encoded using the CLIP ViT-L/14 [44] backbone. For text-video alignment, we calculate the cosine similarity between the video and text prompt, encoded using the ViCLIP-InternVid-10M-Flt [52] checkpoint. For temporal consistency, we calculate the average cosine similarity between features of all adjacent frames, evaluated using two models: CLIP ViT-L/14 [44] and DINOv2 [7].

C.3. TGVE

We also evaluate our PropFly on the TGVE benchmark [55], which contains 76 videos across four editing categories ('style', 'object', 'background', 'multiple'), resulting in a total of 304 editing video-text editing pairs.

For assessing video quality, CLIP temporal consistency, and text-video alignment (ViCLIP$_{out}$), we use the same settings described in EditVerse [23]. Additionally, for text-video direction similarity (ViCLIP$_{dir}$), we calculate the cosine similarity between the text instruction and the video's directional change from the source to the target.

D. Ablation Study

D.1. Details of Ablation studies in main paper

For all ablation studies discussed in the main paper, we trained the models using the same combined dataset (Youtube-VOS [56] and Pexels [1]) for 50,000 iterations, ensuring a fair comparison with our main model.

Full Sampling Baseline. To implement the 'Full Sampling' baseline, we replace our one-step estimation with an iterative solver. Given a video sample $\mathbf{x}_0$, noise $\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a random timestep $t \sim U[0, 1]$, we first obtain the intermediate noised latent $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1$. Instead of a direct one-step estimation, $\mathbf{x}_t$ is denoised via an ODE solver for n steps, where $n = \lceil N \times t \rceil$ and $N = 25$ is the total number of inference steps. This formula ensures that the solver performs the appropriate number of denoising steps required to traverse the trajectory from the current time t down to 0.

We perform this iterative denoising independently using two CFG scales, $\omega_L = 1.0$ and $\omega_H = 7.0$, yielding the fully sampled latents $\hat{\mathbf{x}}_0^{\text{low}}$ and $\hat{\mathbf{x}}_0^{\text{high}}$. These are then used as the source and target training pair, with all other settings identical to PropFly. However, as visualized in Fig. 6, because the two sampling paths are independent, they accumulate numerical errors differently. Consequently, while the resulting videos align well with their text prompts, they frequently diverge in terms of motion structure, leading to severe motion misalignment between the source and target.

Standard FM For the 'Standard FM' baseline, we train the adapter using the standard flow matching objective rather than our proposed GMFM loss. Unlike Eq. 5, we sample new noise $\mathbf{x}'_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and time step $t' \sim U(0, 1)$, and interpolate the target (edited) latent $\hat{\mathbf{x}}_{0|t}^{\text{high}}$ with noise $\mathbf{x}'_1$, and generate new intermediate noised latent $\mathbf{x}'_{t'} = (1-t')\hat{\mathbf{x}}_{0|t}^{\text{high}} + t'\mathbf{x}'_1$. The adapter predicts the velocity using newly generated noisy latent as below:

$$\hat{\mathbf{v}}_{\theta,\phi} = \mathbf{v}_{\theta,\phi}(\mathbf{x}'_{t'}, t', \mathbf{c}_{\text{aug}}, \hat{\mathbf{x}}_{0|t}^{\text{low}}, \hat{\mathbf{x}}_{0|t}^{\text{high}}[0]). \tag{7}$$

The adapter is then trained with original flow matching loss as provided in Eq. 1. As discussed in the main paper, this standard objective guides the model to reconstruct the input video content. However, this approach fails to explicitly learn the transformation mapping required to apply the edit from the first frame to the source structure. Consequently, the 'Standard FM' baseline fails to effectively propagate the edits, often reverting to the original unedited content.

D.2. CFG variation

We further analyze the effect of the CFG scale modulation used during our on-the-fly data pair generation. As described in the main paper, the semantic gap between the low-CFG (ω_L) and high-CFG (ω_H) latents guides the adapter's learning. Here, we fix the low scale at $\omega_L = 1.0$ and vary the high scale ω_H to study its impact on the PropFly model. We compare the video editing performance of models trained using different ω_H values in Table 5.

As shown in Table 5, we observe that a value of $\omega_H = 7.0$ yields the best overall performance. A smaller scale (e.g., $\omega_H = 2.0$) results in lower text alignment, likely because the semantic gap between ω_L and ω_H is too small, providing an insufficient supervision signal for the adapter. Conversely, a very large scale (e.g., $\omega_H = 20.0$) also leads to a slight drop in video quality (Pick Score). This suggests that while the semantic gap is large, the high-CFG latent $\hat{\mathbf{x}}_{0|t}^{\text{high}}$ may begin to contain artifacts or over-saturation, providing a noisy target. Crucially, the model demonstrates **robust and high-quality performance across a stable range of ω_H values** (e.g., 5.0 and 7.0). This indicates that our method is not sensitive to a specific hyperparameter, but rather succeeds as long as it is provided with a clean, strong semantic signal. Our choice of $\omega_H = 7.0$ simply represents the optimal point within this stable region, providing the best balance of semantic strength and visual quality for supervision.

Table 5. Impact of CFG scaling on performance metrics during on-the-Fly data pair generation. Best performance for each metric is highlighted in **bold**, and the second best performance is indicated by an underline.

ω_H	Pick Score ↑	Frame ↑	Video ↑	CLIP ↑	DINO ↑
2	19.98	27.24	24.57	98.83	98.41
5	<u>20.32</u>	<u>28.33</u>	**25.52**	<u>99.01</u>	98.62
7	**20.35**	**28.37**	<u>25.37</u>	**99.03**	<u>98.63</u>
10	20.27	28.24	25.34	99.00	**98.80**
20	20.19	<u>28.33</u>	25.27	98.85	98.62

D.3. On-the-fly Data Pair Quality

Table 6. Synthetic video data quality comparison. We evaluate the text alignment and motion alignment. ↑ indicates higher is better.

Target Videos	Text Alignment ↑	Motion Alignment ↑
Señorita-2M [63]	20.41	78.08
On-the-fly (Ours)	**20.97**	**82.57**

PropFly relies on synthetic source-target latent pairs generated via the pipeline described in Sec. 4.3. In Fig. 7, we provide examples of the on-the-fly synthesized data pairs during training. From the 'Input' videos, the 'Source' and 'Target' videos are synthesized using the one-step clean latent estimation with the low and high CFG values, respectively, during the on-the-fly data pair generation process. As shown, the transformations between the source and target videos cover the local modification and the global transformation. Though the visual quality of the on-the-fly data may not seem optimal (as shown in Fig.7), the difference between the low-CFG and high-CFG samples provides sufficient signal for the model to learn propagation-based video editing, as demonstrated by our video editing results.

To better understand where this ability is coming from, in Table 6, we quantitatively assess the quality of decoded

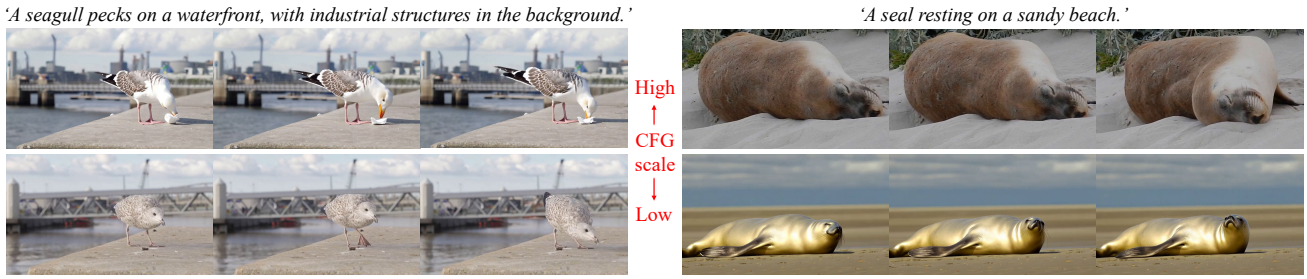

Figure 6. Samples of full sampling with varying CFG scales.

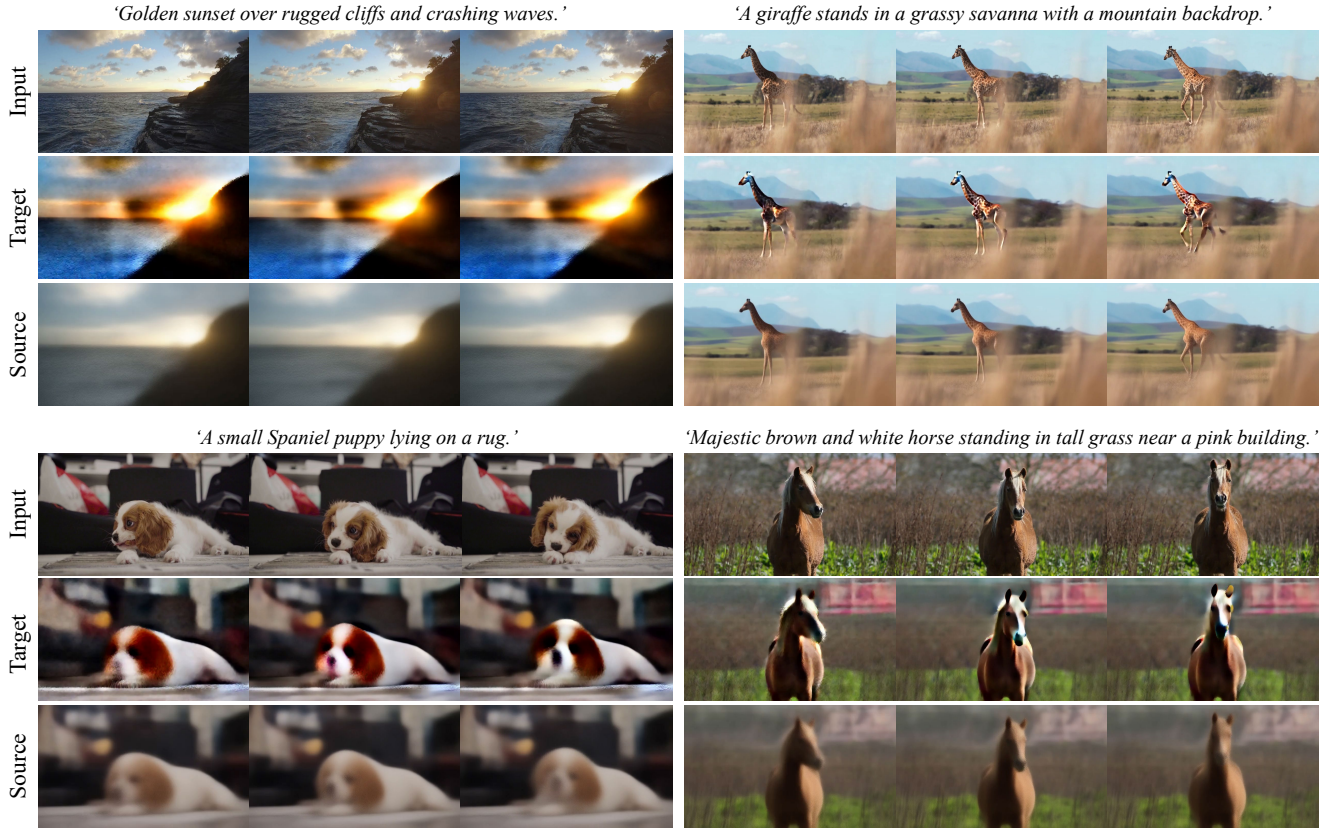

Figure 7. Samples of input videos and their on-the-fly data pairs.

video samples produced by our generation process. We focus on two critical dimensions for video editing training:

- **Text Alignment:** Measures how well the target video reflects the style prompt. We compute the average cosine similarity between the synthesized target videos and their corresponding prompts, encoded using ViCLIP-InternVid-10M-Flt [52].
- **Motion Alignment:** Measures how well the motion structure is preserved between the source and target. We utilize the motion fidelity score proposed in STDF [58], which averages the correlations between tracklets of the source and edited videos, estimated using CoTracker [25].

For comparison, we evaluate 1,000 videos randomly sampled from the 'style transfer' subset of the Señorita-2M dataset [63], which serves as a representative baseline for offline paired training data. As shown in the Table 6, our on-the-fly generated data exhibits superior text and motion alignment, validating the effectiveness of our CFG-based generation strategy.

D.4. Generalization to Other Backbones.

To demonstrate that PropFly utilizes a model-agnostic training strategy, we apply our method to another pre-trained generative backbone: LTX-Video-2B [15]. Similar to our

Table 7. Generalization across backbones. We evaluate the generalization capability of PropFly by applying it to both Wan [50] and LTX-Video [15], measured on the EditVerseBench-Appearance subset [23].

Backbone	Pick Score ↑	Frame ↑	Video ↑	CLIP ↑	DINO ↑
LTX-2B	19.87	27.37	24.68	99.02	98.61
Wan-1.3B	20.35	28.37	25.37	99.03	98.63
Wan-14B	20.42	28.71	26.05	99.21	99.05

main implementation, we attach a VACE adapter to this backbone. However, since we utilize the pre-trained Image-to-Video (I2V) version of LTX-Video, we do not rely on pre-trained adapter weights (which are typically used to turn T2V models into I2V). Instead, the VACE adapter for LTX-Video is **trained from scratch**. As shown in Table 7, the LTX-2B model is successfully trained to perform propagation-based global video editing, achieving high-quality results. While the absolute performance varies with the inherent strength of each backbone and its corresponding VAE, these results demonstrate that our distillation pipeline can be broadly adopted to equip various video generation models with propagation-based capabilities.

E. Limitations & Discussions

While PropFly demonstrates robust performance in propagation-based video editing, it is subject to certain limitations. First, since our method leverages the generative priors of a pre-trained T2V backbone, the resulting video quality and motion dynamics are naturally influenced by the native generative capacity of the base model. It should be noted that this also provides a practical scalability advantage. As stronger T2V backbones are developed, our PropFly can directly benefit from them simply by replacing the underlying model, without modifying the major propagation pipeline itself.

Second, although our approach eliminates the need for costly offline dataset construction, the on-the-fly data pair generation introduces a modest computational overhead during training due to the additional sampling steps. Note that this pipeline allows the supervision distribution to be adapted to changes in the backbone or prompt design, while maintaining the inference-time efficiency of the overall framework.

Finally, our current training pipeline relies on descriptive text guidance (e.g., "A panda is walking") rather than direct edit instructions (e.g., "Change the bear to a panda"). Although this limits the ability to perform edits based purely on text instructions (e.g., instructive V2V), it allows the model to leverage strong visual guidance from the edited first frame. This trade-off results in superior control over content preservation and temporal consistency in the video propagation setting.

Despite these limitations, we believe our PropFly provides a simple and scalable foundation for propagation-based video editing without paired training data, and can serve as a strong basis for more general video editing frameworks.

F. Further Qualitative Comparison

We provide extensive additional qualitative results to further demonstrate the capabilities of PropFly.

F.1. Qualitative Comparison on DAVIS

In Figs. 8 and 9, we present a visual comparison on the DAVIS dataset [41], contrasting PropFly with leading propagation-based methods: AnyV2V [30] and Señorita-2M [63]. As shown, AnyV2V often struggles to perform complex edits that require transforming both the object and the background simultaneously. Similarly, Señorita-2M [63] frequently fails to propagate the transformation consistently or preserve the context of the original videos. In contrast, our PropFly generates high-fidelity videos that faithfully propagate the target transformation while preserving the integrity of the source context.

F.2. Qualitative Comparison on EditVerseBench

We also provide comprehensive qualitative comparisons on the EditVerseBench [23]. We compare against a wide range of baselines, including InsV2V [9], LucyEdit [50], STDF [58], VACE [22], EditVerse [23], Runway [46], and Señorita-2M [63], utilizing their publicly available results.

To ensure a fair comparison, we adopt specific protocols for the input conditions depending on the task type. For the propagation tasks shown in Fig. 10, we employ the *same* provided edited first frame for PropFly and other propagation-based baselines [30, 63]. One exception is CCEdit [12], where we utilize the edited *center* frame from the EditVerse baseline to accommodate its bidirectional propagation mechanism. For general editing tasks (Figs. 11 and 12), such as object modification, addition, or removal, we utilize the edited first frame generated by EditVerse [23] as the starting condition for our propagation.

Fig. 10 shows the video propagation comparison. We observe that text-guided video editing methods such as STDF [58], LucyEdit [50], and InsV2V [9] fail to propagate the specific transformed style, as they rely solely on text instructions. Other methods [12, 22, 23, 30, 46, 63] demonstrate propagation capabilities, they often struggle to reconstruct complex dynamics, such as the fast motion of the bird. In contrast, our PropFly successfully propagates the transformed style across the entire video while faithfully preserving the original motion. In Fig. 11, other propagation methods [12, 30, 63] often fail to consistently propagate the woman's changed clothes or accurately reconstruct her

motion. On the other hand, our PropFly robustly maintains the edited appearance throughout the sequence and successfully preserves the fidelity of the subject's movement.

We also compare the object addition and removal quality across methods in Fig. 12. For object addition (left side of Fig. 12), other propagation-based methods [12, 30, 63] struggle to synthesize the girl's motion naturally, whereas our PropFly generates faithful and coherent motion. For object removal (right side of Fig. 12), while other baselines [12, 30, 63] fail to plausibly fill the removed region, our PropFly effectively synthesizes the girl's left hand, maintaining temporal consistency. It is worth noting that PropFly is not explicitly trained for object addition or removal tasks, nor does it utilize mask guidance. However, these results demonstrate that our model learns a sufficiently robust and generalized transformation to handle such complex structural edits effectively.

Figure 8. Video quality comparison between propagation-based video editing methods on the DAVIS dataset [41].

Figure 9. Video quality comparison between propagation-based video editing methods on the DAVIS dataset [41].

'A small robin with a bright orange chest perches on a mossy log, looking around alertly. After a moment, the bird spreads its wings and quickly flies away, leaving the log empty and peaceful.'

→ *'A small robin with a bright orange chest stands still on a mossy log, glancing alertly from side to side. The bird remains perched, and the tranquil forest setting is uninterrupted, capturing a peaceful moment frozen in time.'*

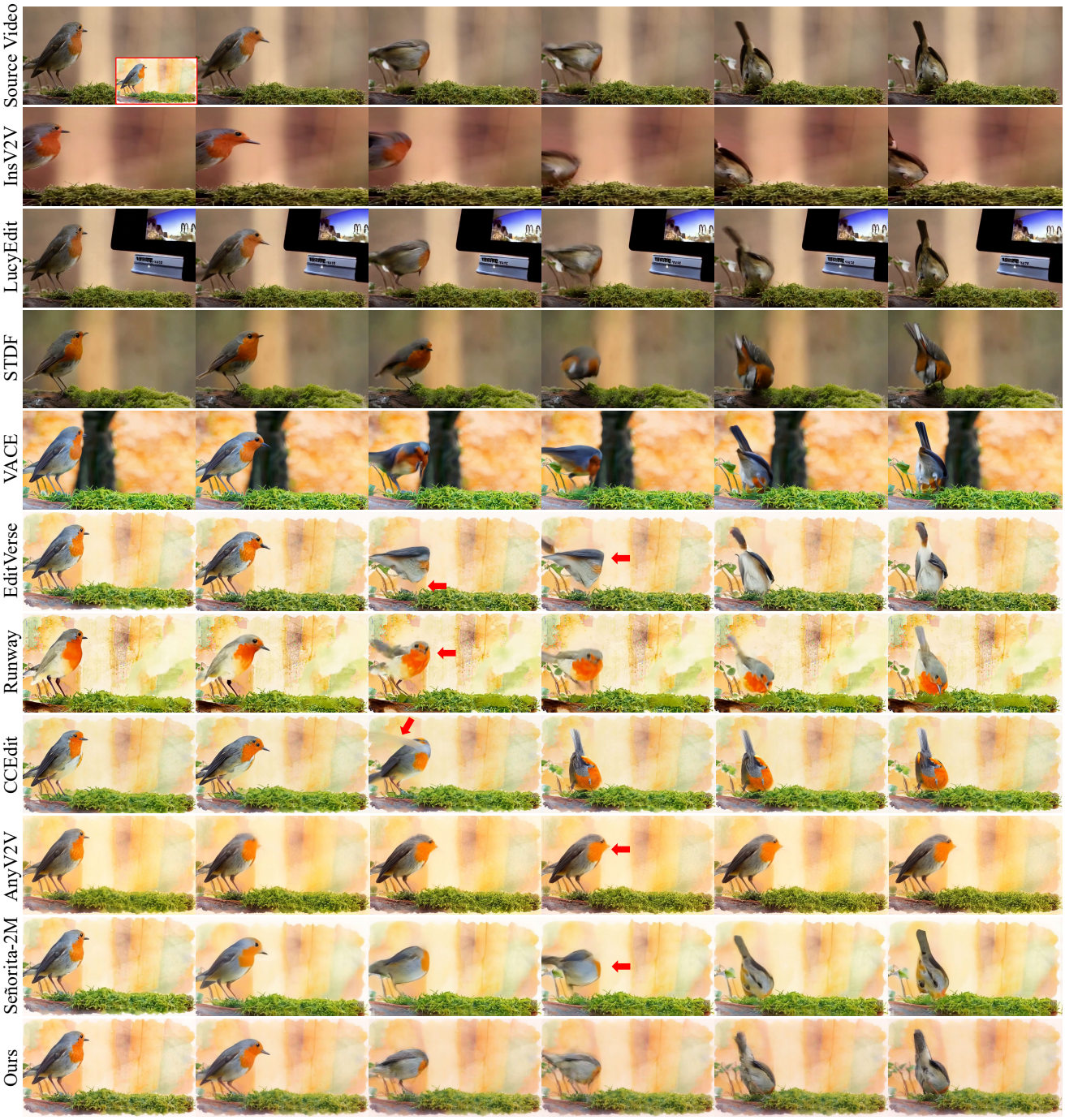

Figure 10. Video quality comparison on the **Propagation** task in EditVerseBench [23]

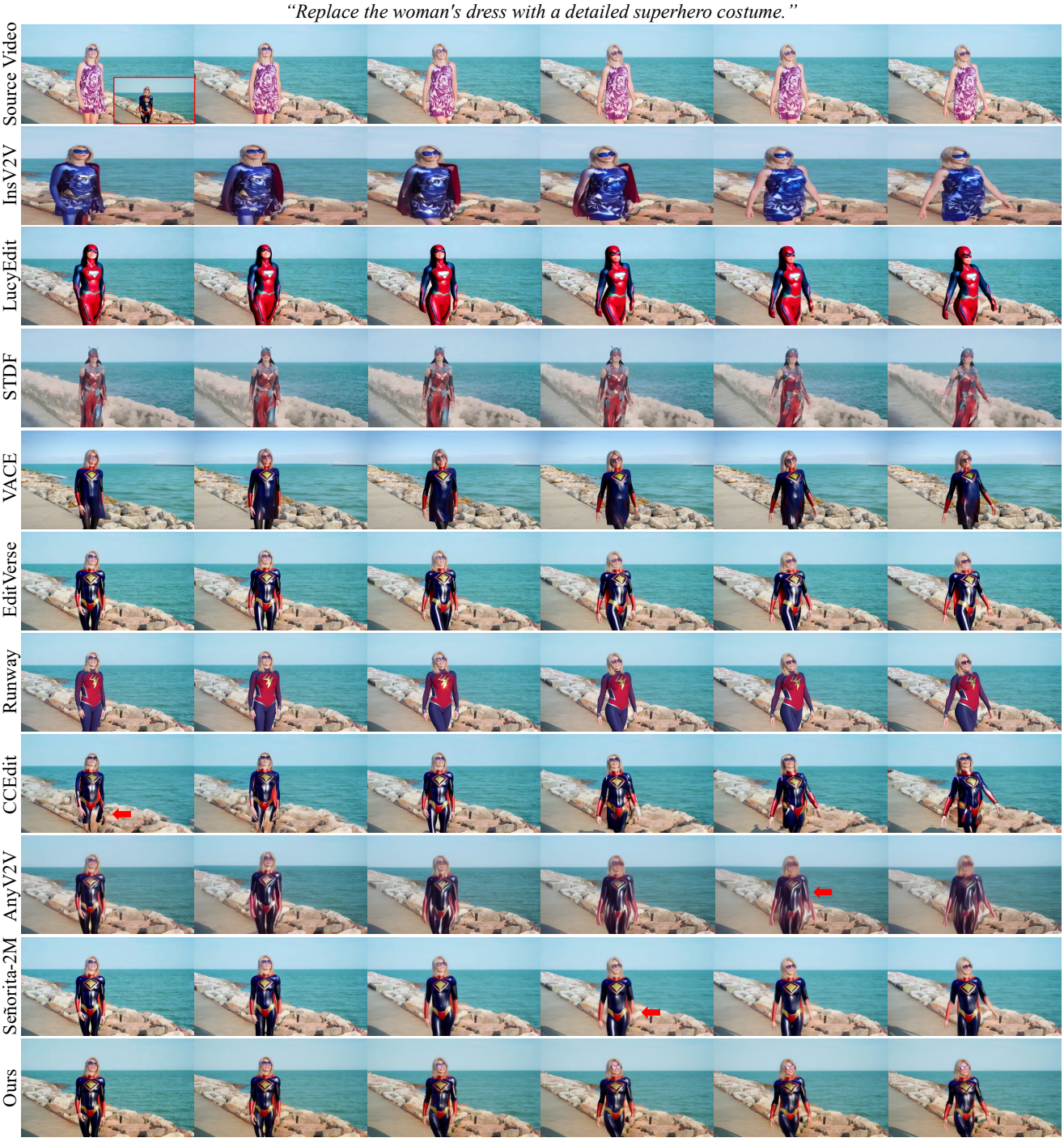

Figure 11. Video quality comparison on the **Change object** task in EditVerseBench [23]

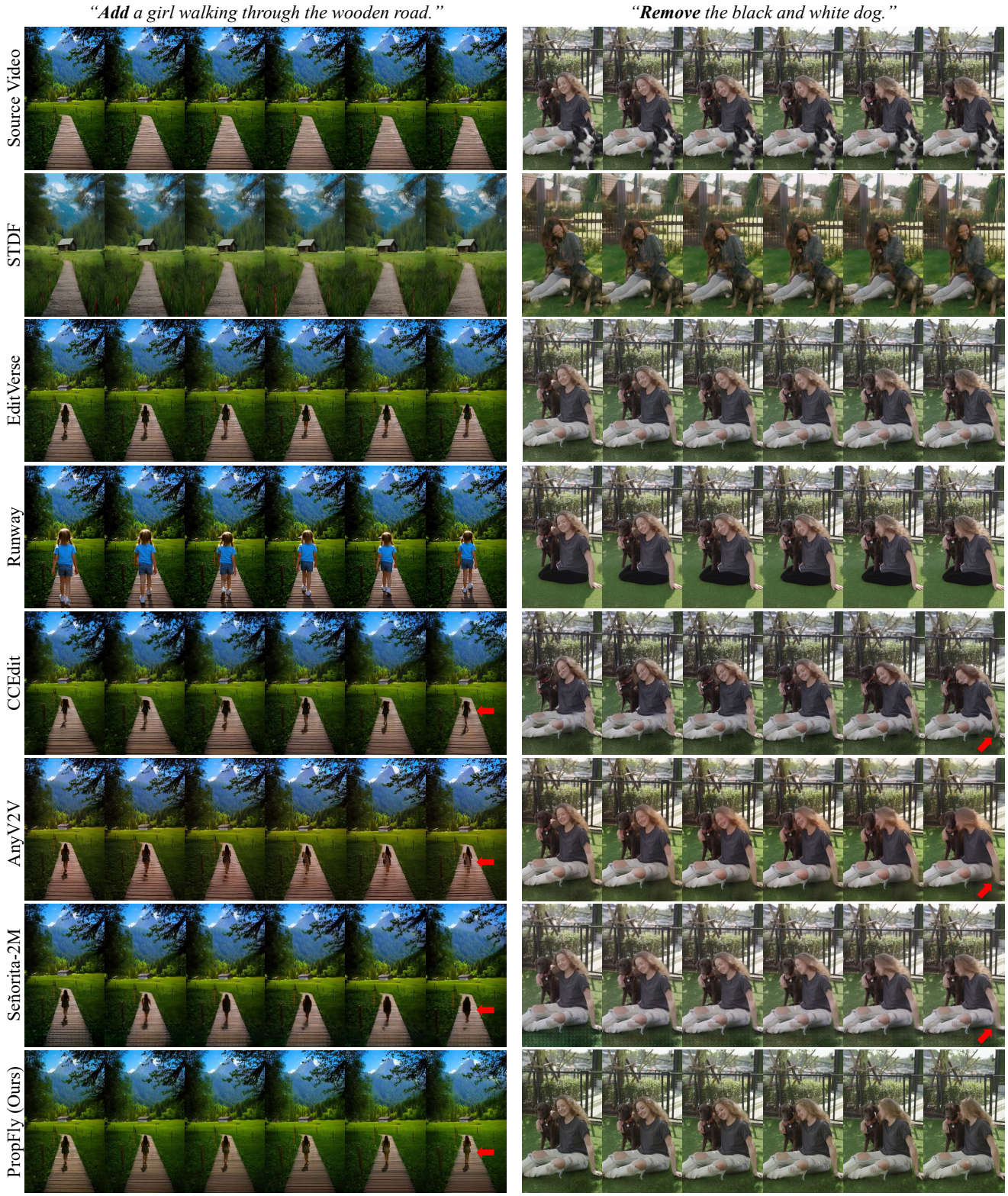

Figure 12. Video quality comparison on the **Add object** and **Remove object** tasks in EditVerseBench [23]