ROLLOUTREMOVER: SCRIBBLE-GUIDED OBJECT REMOVAL VIA LATENT ROLLOUT MODELING

Anonymous authors
Paper under double-blind review

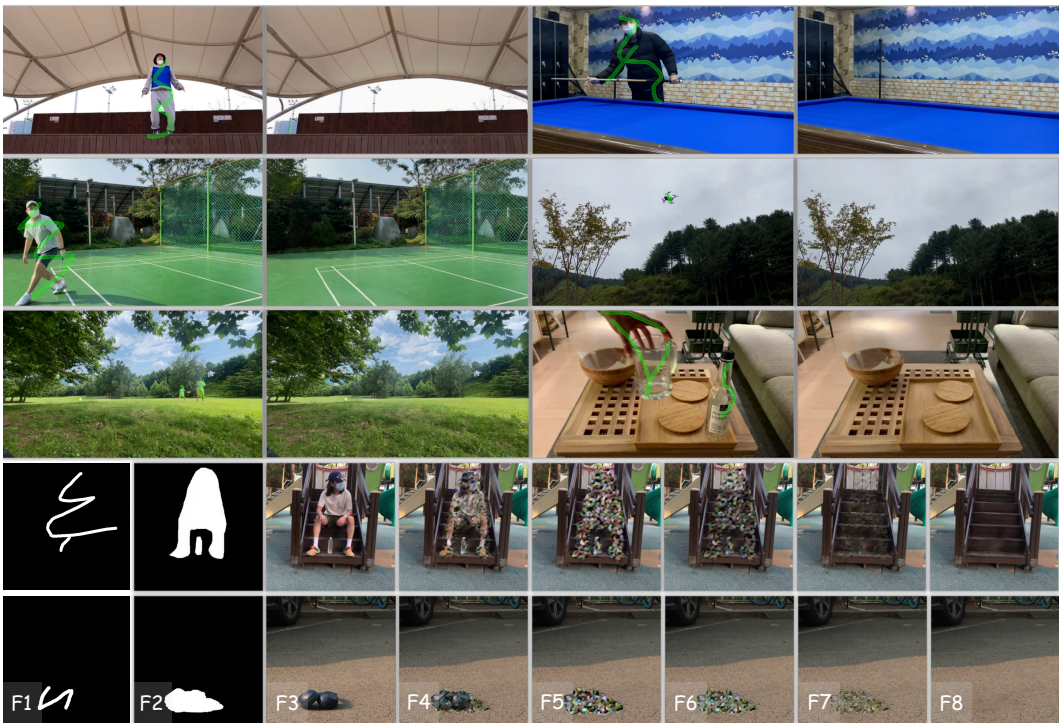


Figure 1: The top three rows show object removal results from scribble masks, and the bottom two rows show explicit rollouts for object corruption and background restoration generated by video diffusion models. F1 is the user scribble mask, F2 is the inferred removal region, and F3–F8 show the transition from source image to object-removed result. For visualization, the rollout is subsampled from the full 14-frame generation sequence.

ABSTRACT

Object removal aims to remove target objects and recover the background they occlude. Existing methods have achieved strong visual fidelity, but they still rely on mask-conditioned inpainting and therefore require an accurate object mask as input. In practice, however, users often provide only a casual scribble, while current pipelines either rely on limited scribble generalization or introduce external models to bridge the gap. We present **RolloutRemover**, a video diffusion framework for scribble-guided object removal. Instead of directly predicting the final removed image, RolloutRemover interprets the scribble as removal intent and performs removal through a structured latent rollout that explicitly disentangles object corruption from background restoration, providing stronger supervision over the removal process. We also introduce **SGORBench** with multiple mask sets for comprehensive evaluation of image object removal. Experiments show that RolloutRemover achieves strong removal quality and is substantially more robust to sparse user-provided scribbles than prior mask-dependent pipelines.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Object removal seeks to eliminate a target object from a scene and restore the underlying background, producing an image as if the object were never present. Recent diffusion-based methods, such as OmniPaint (Yu et al., 2025), ObjectClear (Zhao et al., 2026), and OSOR (Zhou et al., 2026), have substantially improved removal fidelity, especially in handling challenging object effects like shadows and reflections. However, despite their strong performance, these methods largely formulate object removal as mask-conditioned inpainting, assuming that the target removal region is already accurately specified.

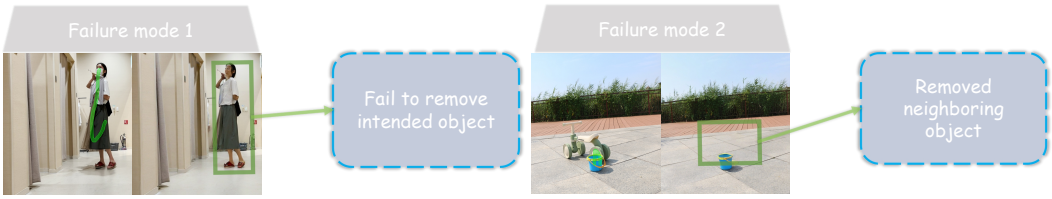


Figure 2: Two failure modes of scribble-guided object removal. The left example shows the prior method OmniPaint failing to remove the object due to an incomplete scribble, while the right example shows unintended removal of a neighboring object caused by an over-extended scribble.

In practice, users often provide only a casual scribble to indicate the object they wish to remove, rather than a precise object mask. This creates a mismatch between realistic user interaction and the assumptions made by existing methods. Although some recent methods attempt to adapt to scribble inputs by training with degraded masks, they often fail to accurately capture scribble intent, as illustrated by the two failure modes in Figure 2: over-extended scribbles may cause unintended removal of neighboring content, while partial scribbles may lead to incomplete editing. A casual scribble should therefore be viewed not as a degraded object mask, but as sparse user intent that may cross object boundaries, cover only a discriminative part of the object, or overlap with nearby content. The central challenge is thus not merely inpainting a specified region, but inferring the intended removal region from sparse user input.

Due to the sparse nature of scribble masks, transforming user input into the exact removal region requires semantic understanding and contextual reasoning. A model must reason beyond the pixels directly marked by the scribble: it should identify the complete object implied by the scribble, distinguish it from nearby content, and infer the background that should appear after removal. Video diffusion models provide a sequence representation in which we can jointly model and supervise the removal region, intermediate edit states, and the final image.

However, simply replacing the inpainting backbone with a video diffusion model is still insufficient. A key challenge is that video models do not automatically resolve object removal from a sparse in-context sequence. If the final target image is directly appended after the source frame, the model must bridge an abrupt endpoint shift from object present to object free between two consecutive frames. This creates an endpoint-supervision shift and places a heavy burden on the model's in-context visual reasoning ability and is poorly matched to the inductive bias of video diffusion models, which are generally better suited to gradual and structured temporal variation. A natural solution is to design a gradual removal process that conducts object removal in a slow transition. In particular, we construct a noise-based rollout in latent space that decomposes removal into a corruption and restoration process. The corruption phase progressively destroys the visual evidence of the target object, while the restoration phase recovers the underlying background. Moreover, latent space noise is naturally compatible with diffusion dynamics, enabling explicit supervision over the removal process and encouraging gradual rather than discontinuous transformation.

Building on these observations, we propose RolloutRemover, a video-diffusion-based framework for scribble-guided object removal that directly interprets casual scribbles as removal intent and performs object removal through an explicit latent rollout separating object corruption from background restoration. We also introduce a new benchmark with three forms of masks: a casual scribble mask, an object mask, and an object-effect mask, enabling systematic evaluation of final removal quality, robustness to free-form user input, and handling of object-associated effects.

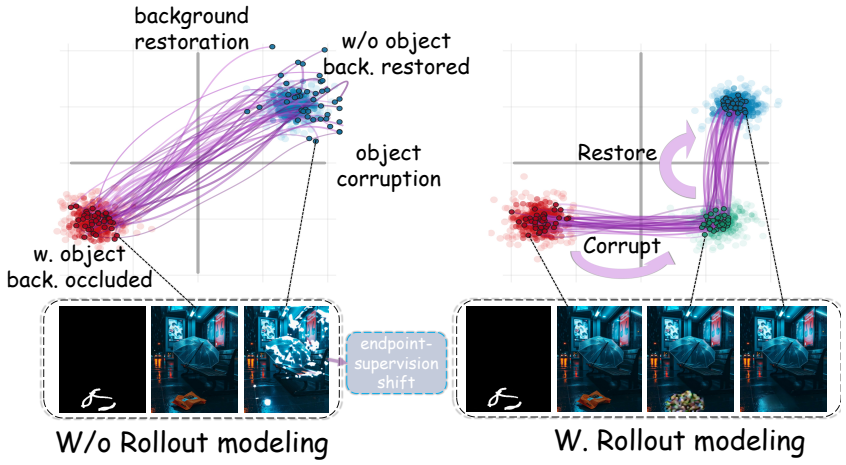


Figure 3: A qualitative illustration of object-background disentanglement in object removal, where object corruption and background restoration define two orthogonal axes. The left side illustrates the endpoint-supervision shift without rollout modeling, which yields poor results, whereas the right side demonstrates our rollout modeling, which leads to substantially improved outcomes.

Our contributions are three-fold:

- We propose RolloutRemover, a framework for scribble-guided object removal that can infer the intended removal region from casual user scribbles, reducing the dependence of prior methods on precise object region specification.
- We introduce an explicit latent rollout design for object removal, where a noise-based corruption and restoration trajectory provides stronger process supervision and improves editing quality.
- We introduce SGORBench with scribble, object, and object-effect mask sets, enabling comprehensive evaluation of scribble-guided object removal in terms of removal quality, robustness to imperfect user input, and handling of object-associated effects.

2 Related Work

2.1 Object Removal and Image Inpainting

Object removal is typically formulated as a mask-conditioned inpainting problem, where the model reconstructs the background after removing the target object. Generic inpainting models (Black Forest Labs, 2024; Zhang et al., 2025) provide strong priors for filling masked regions, and recent image-based removal methods, such as ObjectClear (Zhao et al., 2026), OmniPaint (Yu et al., 2025), OSOR (Zhou et al., 2026), and OmniEraser (Wei et al., 2025), further improve removal quality, especially for object-associated effects such as shadows and reflections. However, these methods still rely on accurate object masks as input.

2.2 Video Models for In-Context Visual Reasoning and Editing

Video models have demonstrated strong spatial understanding and reasoning capabilities (Ma et al., 2026; Gu et al., 2025; Wiedemer et al., 2025). Recent video models effectively exploit structured multi-frame context for visual reasoning and transformation. VideoCoF (Yang et al., 2025) reasons about editing regions during generation, while ChronoEdit (Wu et al., 2025) reformulates image editing as temporal generation with intermediate states between source and target images. Related work such as GeoRelight (Xue et al., 2026) further suggests that multi-frame conditioning can improve

visual understanding and structured generation. Our work extends this paradigm to scribble-guided object removal with explicit supervision of a two-phase latent rollout. The rollout passes through a noise anchor that replaces the removal region with Gaussian noise, separating object corruption from background restoration.

2.3 Scribble-Guided Editing and Interactive Removal

Scribbles, sketches, and rough brush masks have been explored as lightweight interaction signals in image editing (Zeng et al., 2021; Ji et al., 2026). However, many existing methods still rely on localized inpainting or mask-conditioned generation, such as BrushNet (Ju et al., 2024) and SmartEraser (Jiang et al., 2025), and therefore remain sensitive to free-form user scribble indications. Interactive pipelines such as Inpaint Anything (Yu et al., 2023) first obtain masks through external segmentation and then perform inpainting. In contrast, our method directly treats a casual scribble as removal intent, without requiring it to precisely or fully cover the target object.

3 Methodology

Given an input image containing a target object and a user-provided scribble indicating removal intent, our goal is to generate an object-removed result without requiring a precise object mask at inference time. Prior object removal methods typically formulate the task as direct endpoint prediction from an input image and a complete object mask. In contrast, our method performs removal through a structured multi-frame latent generation process conditioned on the user scribble.

The generated sequence contains several types of frames with different roles. It starts from a scribble-intent frame that represents the user's coarse indication, and includes a removal-region frame that captures the full spatial extent of the desired removal inferred from the scribble and source image. The source-image frame provides the original scene, while a set of intermediate rollout frames models the gradual removal process before reaching the final target frame corresponding to the object-free result. During training, the removal-region frame and the rollout sequence are supervised. For scribble-guided inference, the model takes only the input image and the user scribble as input.

3.1 Overview of RolloutRemover

RolloutRemover operates on a multi-frame latent sequence that turns scribble-guided object removal into a structured generation problem. In the full sequence

$$\mathcal{S} = \{f_1, f_2, f_3, \mathcal{R}, f_N\}, \tag{1}$$

the first frame f_1 is a scribble-intent frame that represents the user's coarse indication of what should be removed. It is followed by a removal-region frame f_2 that captures the full spatial extent of the desired removal, including both the target object and its associated effects. The next frame f_3 is the source-image frame, which provides the original scene context. Starting from this frame, the model predicts a sequence of intermediate rollout frames $\mathcal{R}$ that describe how the target content disappears and how the background is gradually recovered, before finally producing the target frame f_N corresponding to the object-free result.

Different parts of the sequence play different roles. The scribble-intent frame specifies user intent, the removal-region frame makes the required removal extent explicit, and the source-image frame anchors the scene identity. The rollout frames bridge the source and target states through a structured removal process, rather than forcing the model to jump directly from the original image to the final edited result. In this way, the model is trained not only to generate the final object-free image, but also to model how the removal should unfold over time.

3.2 Removal Rollout Construction

The key component of RolloutRemover is the explicit removal rollout inserted between the source and target states. We construct this rollout in latent space, where all frames are represented in the same visual space used by the underlying diffusion model. Compared with directly supervising a

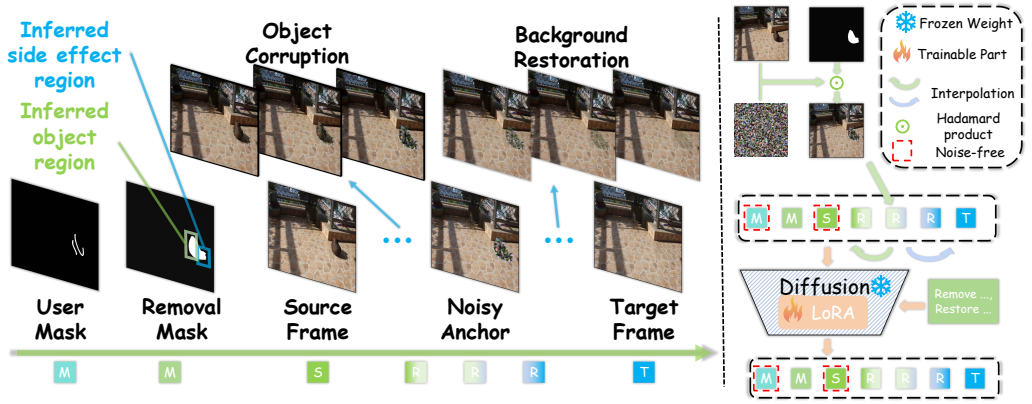


Figure 4: Overview of RolloutRemover. Given a user scribble and the source image, the model jointly generates a removal mask covering the object and side effect regions, intermediate rollout frames, and the final target frame. The latent rollout transitions from the source frame to a noisy anchor and finally to the target frame, separating object corruption from background restoration. The right panel illustrates the training setup with frozen and trainable components.

source-to-target jump, the rollout provides explicit intermediate supervision for the removal process and makes the transition between object-present and object-free states more gradual.

Let $z_s = E(x_s)$ and $z_t = E(x_t)$ denote the source and target latents encoded by the VAE encoder $E(\cdot)$. Let $m_r \in \{0,1\}^{h\times w}$ denote the removal-region mask at latent resolution, covering both the target object and its associated effects. Our goal is to construct a sequence of intermediate latents

$$\mathcal{R} = \{z_k\}_{k=1}^{K}, \qquad K = K_c + K_r + 1, \tag{2}$$

where K_c and K_r are the numbers of corruption and restoration frames, respectively, and the extra frame is the noise anchor.

Our rollout follows a two-stage design. The first stage progressively corrupts the target region, making the disappearance of the object explicit. The second stage restores the underlying background, so that the model learns not only what should be removed, but also how the missing content should be recovered. This decomposition matches the intrinsic structure of object removal more closely than direct endpoint prediction, where object disappearance and background restoration are combined into a single step.

To realize this idea, we first construct a noise anchor in latent space. Let $\epsilon \sim \mathcal{N}(0, I)$ be Gaussian noise with the same shape as z_s. The anchor is defined as

$$z_a = (1 - m_r) \odot z_s + m_r \odot \epsilon, \tag{3}$$

where $\odot$ denotes element-wise multiplication. In this way, the unmasked region preserves the source scene, while the removal region is replaced with noise. The anchor serves as an information-neutral midpoint between the source object and the final clean background.

Based on this anchor, we build the rollout in two phases. In the corruption phase, the latent state progressively moves from the source latent toward the noise anchor:

$$z_i^{\text{cor}} = (1 - \alpha_i) z_s + \alpha_i z_a, \qquad i = 1, \ldots, K_c, \tag{4}$$

where

$$\alpha_i = \left(\frac{i}{K_c + 1}\right)^{\gamma}. \tag{5}$$

In the restoration phase, the latent state progressively moves from the noise anchor toward the target latent:

$$z_j^{\text{res}} = (1 - \beta_j) z_a + \beta_j z_t, \qquad j = 1, \ldots, K_r, \tag{6}$$

where

$$\beta_j = 1 - \left(1 - \frac{j}{K_r + 1}\right)^{\gamma}. \tag{7}$$

The complete rollout is defined as the ordered sequence of the corruption states, the noise anchor, and the restoration states,

$$\mathcal{R} = \{z_i^{\mathrm{cor}}\}_{i=1}^{K_c} \cup \{z_a\} \cup \{z_j^{\mathrm{res}}\}_{j=1}^{K_r}. \tag{8}$$

This construction has several advantages. It explicitly separates object disappearance from background recovery, which provides stronger supervision over the removal process. Corruption preserves source context outside the removal region, while restoration approaches the target, helping maintain a stable scene identity. Finally, performing the rollout in latent space avoids repeated pixel-latent conversion and keeps the intermediate states aligned with the diffusion model's native dynamics. In this way, the rollout is not treated as an unconstrained intermediate trajectory, but as a removal-specific process tailored to the structure of object removal itself.

3.3 Training and Inference

Specifically, the scribble-intent frame f_1 and the source-image frame f_3 are treated as condition frames and remain clean throughout training. To improve robustness to different forms of mask guidance, we uniformly sample one of a scribble mask, an object mask, or an object-effect mask with equal probability to construct the user-intent frame f_1. The predicted frames include the removal-region frame f_2, the rollout frames in $\mathcal{R}$, and the final target frame f_N. In this way, the model is supervised not only on the final object-removed result, but also on the intermediate reasoning and transition process that leads to it.

We adopt a standard flow-matching training objective (Lipman et al., 2023) over the predicted frames. During training, noise is added only to the frames that need to be generated, while the condition frames remain unchanged and do not contribute to the loss. The model therefore learns to reconstruct the full removal sequence from fixed source context and user intent. This training strategy encourages the model to jointly infer the desired removal extent from the scribble frame and model how the target content disappears and how the background is recovered through the rollout frames.

At inference time, the source image and supplied condition mask are encoded as clean frames f_3 and f_1, respectively. All other frames, including the removal-region frame f_2, are initialized from Gaussian noise and jointly denoised. The object-removed result is decoded from f_N.

4 Experiments

4.1 Experimental Setup

Training data. We collect training data from publicly available object removal datasets, including OBER (Zhao et al., 2026) and CORNE (Zhou et al., 2026). Specifically, we select high-quality real-world samples from OBER and incorporate synthetic samples from CORNE as complementary data. The resulting training pool contains 50K samples.

Benchmark construction. We construct SGORBench by curating 300 samples from OBER, RORD-Val and CORNE. Although the training set and SGORBench are drawn from the same upstream data collections, we enforce disjoint training and evaluation splits. Specifically, we remove exact case overlaps and further exclude perceptually same-scene cases through image similarity and manual verification. Each sample has three aligned masks: scribble, object, and object-effect (see Figure 1 in Appendix D). For each sample, we retain the original object mask as the reference region for quantitative evaluation.

Implementation details. We build RolloutRemover on the Wan2.1-T2V-1.3B video diffusion backbone (Wan et al., 2025) and fine-tune it with LoRA (rank 128) on our training split. All frames are processed at 480×832 pixels, corresponding to a 60×104 latent grid. We use the AdamW optimizer (Loshchilov & Hutter, 2019) with a learning rate of 1×10^{-4}, weight decay 3×10^{-2}, ϵ=10^{-10},

and a maximum gradient norm of 0.05. Training uses bf16 mixed precision with DeepSpeed ZeRO-2 and an effective batch size of 8. The rollout is constructed with K_c=4 corruption frames, K_r=5 restoration frames, and interpolation exponent γ=1.2, yielding a total sequence length of 14 frames. At inference time, we use 50 denoising steps with classifier-free guidance scale 5.0. Although the Wan2.1 backbone employs a 3D causal VAE, we decode each latent frame independently to obtain per-frame output images.

Baselines. We compare RolloutRemover with five mask-conditioned methods: OmniPaint, ObjectClear, OSOR, SmartEraser, and OmniEraser (Yu et al., 2025; Zhao et al., 2026; Zhou et al., 2026; Jiang et al., 2025; Wei et al., 2025). In each main SGORBench setting, all methods use the same mask and resolution per sample. Appendix F also tests the baselines with SAM-derived masks.

Metrics. We report standard image quality metrics, including PSNR, SSIM, LPIPS, FID, CFD and ReMOVE (Wang et al., 2004; Heusel et al., 2018; Zhang et al., 2018; Chandrasekar et al., 2024; Yu et al., 2025). We also include human evaluation in the form of a user study, which assesses removal completeness, background plausibility, edge seamlessness, non-target preservation and overall performance.

Evaluation. We evaluate our method and baselines on the proposed SGORBench under different mask conditions for a comprehensive assessment, and additionally on OmniPaint-Bench to measure general object removal performance.

Table 1: Comparison with prior methods on SGORBench under the **scribble mask** condition. Best results are in **bold**, and second-best results are underlined. All methods are evaluated at the same resolution for each sample.

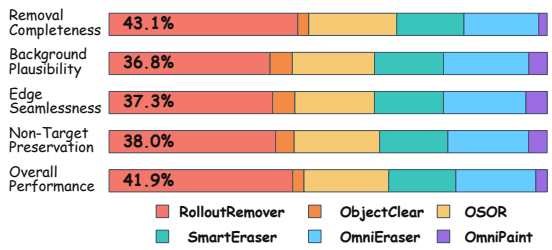
Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
ObjectClear	22.31	0.7818	0.1598	80.64	0.4877	0.7542
OmniPaint	20.78	0.8455	0.1393	102.49	0.5292	0.6193
OSOR	26.38	0.8325	0.1536	58.05	0.3203	0.8905
SmartEraser	21.87	0.7657	0.1891	77.36	0.4432	0.7459
OmniEraser	21.92	0.7530	0.2019	71.26	0.3868	0.8149
RolloutRemover (Ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**

Table 2: Comparison with prior methods on SGORBench under the object-mask and object-effect-mask conditions. Best results are in **bold**, and second-best results are underlined.

	Object Mask						Object-Effect Mask					
Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
ObjectClear	25.81	0.8006	0.1135	30.19	0.2569	**0.9407**	24.70	0.7859	0.1340	35.13	0.2428	0.9411
OmniPaint	27.38	**0.8801**	**0.0697**	**21.43**	0.2450	0.9318	25.99	0.8522	**0.1005**	**30.08**	0.2528	0.9339
OSOR	**28.11**	0.8450	0.1301	38.57	0.2186	0.9375	**28.05**	0.8441	0.1337	39.35	0.2369	0.9306
SmartEraser	23.86	0.7788	0.1635	50.37	0.3120	0.8945	24.65	0.7785	0.1587	41.19	0.3000	0.9197
OmniEraser	23.58	0.7702	0.1761	46.42	0.2740	0.9376	23.31	0.7625	0.1846	48.47	0.2589	0.9408
RolloutRemover (Ours)	27.35	0.8631	0.1327	31.24	**0.2183**	0.9382	26.58	**0.8532**	0.1491	35.53	**0.2011**	**0.9413**

Table 3: Quantitative comparison on OmniPaint-Bench. Best results are in **bold**, and second-best are underlined.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
ObjectClear	22.35	0.7366	0.2751	66.97	0.2212	**0.9332**
OmniPaint	23.48	**0.8178**	**0.1522**	**50.38**	0.2404	0.9192
OSOR	**24.52**	0.8091	0.2019	62.56	0.2108	0.9207
SmartEraser	20.84	0.7265	0.3189	94.37	0.2909	0.8904
OmniEraser	22.05	0.7188	0.2503	75.02	0.2703	0.9274
RolloutRemover (Ours)	23.37	0.8129	0.1627	58.12	**0.2104**	0.9301



(a) Normalized Top-2 selections (%)

Method	RC	BP	ES	NP	Overall
ObjectClear	2.5	5.1	5.1	4.2	2.6
OmniPaint	1.9	4.2	4.8	4.2	2.6
OSOR	20.1	18.8	18.2	19.5	19.3
SmartEraser	15.3	15.8	15.8	15.6	15.3
OmniEraser	17.1	19.5	18.8	18.4	18.2
RolloutRemover (Ours)	**43.1**	**36.8**	**37.3**	**38.0**	**41.9**

(b) Exact normalized shares (%)

Figure 5: Human evaluation results. Panel (a) visualizes each method's normalized share of all Top-2 selections, while panel (b) reports the corresponding exact values. Since each evaluation contributes two selections, the Top-2 inclusion rate is twice the normalized share. Best results are in **bold**, and second-best results are underlined. RC: Removal Completeness; BP: Background Plausibility; ES: Edge Seamlessness; NP: Non-Target Preservation. Overall Performance is the primary criterion.

4.2 Main Results

Overall comparison. Tables 1 and 2 compare RolloutRemover with prior methods on SGOR-Bench. On scribble masks, RolloutRemover clearly outperforms all baselines, demonstrating strong robustness to sparse and imperfect user input. On object and object-effect masks, RolloutRemover approaches the strongest existing methods. We further evaluate on OmniPaint-Bench in Table 3, where RolloutRemover achieves near-SOTA performance. These results show that our method remains competitive with image inpainting methods built on specialized backbones, suggesting that explicitly modeling the removal process and intermediate transitions provides additional benefits beyond strong inpainting priors. The learned temporal rollout enables better interpretation of user intent and progressive background recovery.

Ablation Studies. Please refer to our ablation study in Appendix C.

4.3 User Study

Protocol. We conduct a method-blinded study with 60 participants on 20 randomly selected SGOR-Bench cases. Participants view each scribble-marked source image and six randomly ordered anonymous results, selecting their top two for removal completeness, background plausibility, edge seamlessness, non-target preservation, and overall performance. Study details appear in Appendix I.

Results. As shown in Figure 5, human judgments consistently favor RolloutRemover across all five criteria. For overall performance, the primary criterion, RolloutRemover's Top-2 inclusion rate is 83.8%, versus 38.7% for OSOR, the strongest baseline. These results complement the quantitative metrics, showing our method's perceptible advantages under realistic user evaluation.



Figure 6: Qualitative results of RolloutRemover under scribble guidance. Each group shows the source image with its scribble mask, the associated object mask, the RolloutRemover output and the ground truth.

4.4 Qualitative Results

Qualitative gallery. In Figure 6, RolloutRemover uses scribbles to remove intended objects and their effects across diverse scenes, preserving nearby structures and restoring coherent backgrounds.

Visual comparison. In Figure 7, RolloutRemover removes the intended object more completely than prior pipelines while preserving nearby structures and producing cleaner restored backgrounds. Appendix H shows further comparisons and RolloutRemover failure cases.

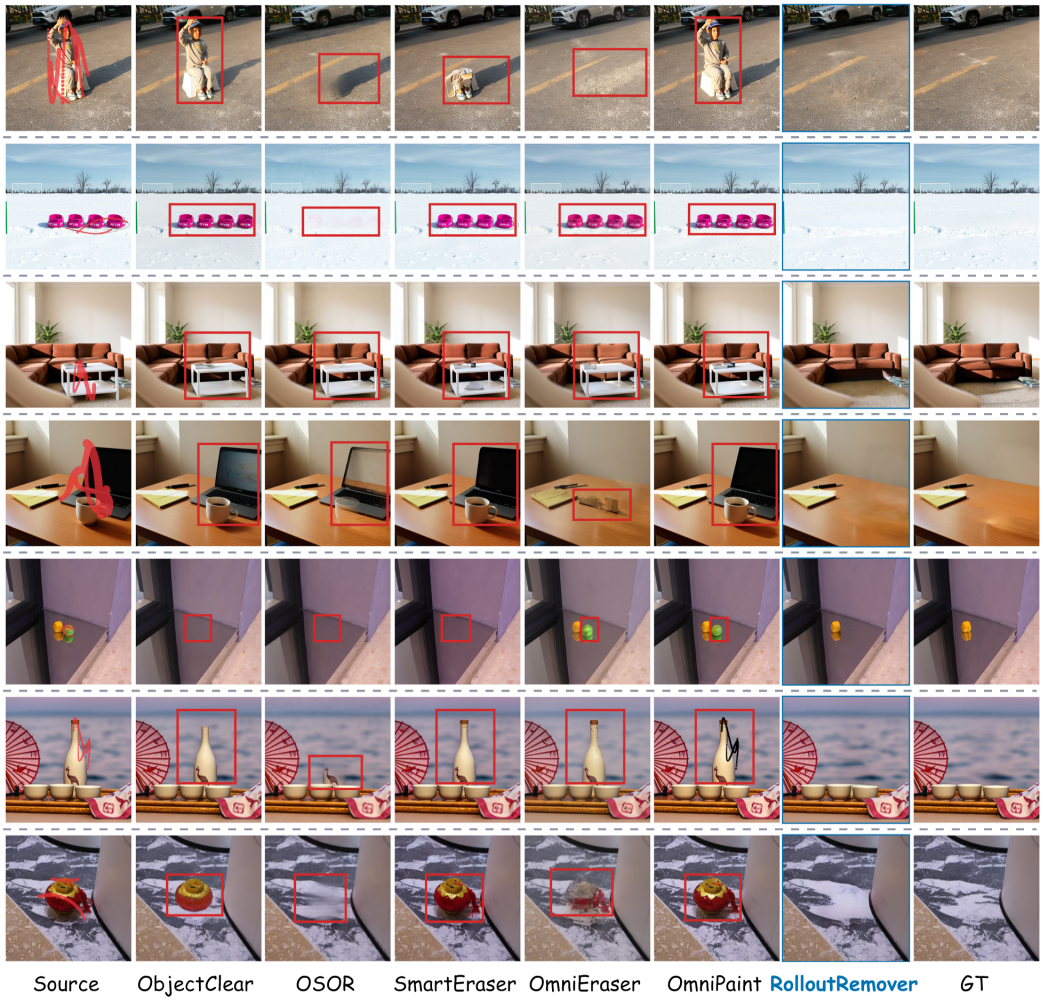


Figure 7: Qualitative comparisons under scribble guidance. Each row shows the source image, outputs from baselines and RolloutRemover, followed by the ground truth. Red boxes highlight incomplete removal, residual objects or unintended artifacts in baseline outputs.

5 Conclusion

We introduced RolloutRemover, a scribble-guided object removal framework that directly interprets casual scribbles as removal intent and avoids the fragile segmentation-dependent pipeline of prior methods. Our key idea is to model removal as an explicit latent rollout, which separates object corruption from background restoration and provides stronger supervision over the editing process. We also presented a new benchmark with scribble, object, and object-effect annotations for comprehensive evaluation. Results show that RolloutRemover delivers strong removal quality and improved robustness under imperfect user input.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing. We did not use generative AI tools to generate experimental results, fabricate data, or make final scientific claims, and other disclosure categories are not applicable to this work. All AI-assisted text was manually reviewed, revised, and verified by the authors. We take full responsibility for the final content of this work, including text and claims produced with the aid of generative AI.

ETHICS STATEMENT

This work studies scribble-guided object removal for research purposes. While such techniques can support benign applications such as photo editing and data augmentation, they may also be misused to manipulate image content in deceptive ways. We therefore do not recommend deployment in high-stakes settings without human oversight. Our evaluation includes a method-blinded user study, for which only aggregate results are reported. We will document the intended use, limitations, and potential misuse risks in any public release.

REPRODUCIBILITY STATEMENT

We make our work reproducible by providing a complete description of the model design, rollout construction, training and inference procedure, and benchmark setup in the main paper and appendix. In particular, Section 3 specifies the task setup, sequence design, latent rollout construction, and training/inference protocol, while the experimental section and supplementary material describe the benchmark annotations, baselines, evaluation settings, and implementation details. We will release code, data processing scripts, benchmark annotations, and configuration files to support reproduction of the reported results.

REFERENCES

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Aditya Chandrasekar, Goirik Chakrabarty, Jai Bardhan, Ramya Hebbalaguppe, and Prathosh AP. Remove: A reference-free metric for object erasure, 2024. URL https://arxiv.org/abs/2409.00707.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3d-aware video diffusion for versatile video generation control, 2025. URL https://arxiv.org/abs/2501.03847.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

Anya Ji, George Ma, Téa Wright, Yiming Zhang, David M. Chan, Alane Suhr, and Somayeh Sojoudi. Scribbleedit: Synthetic data for image editing with scribbles and text, 2026. URL https://arxiv.org/abs/2605.01135.

Longtao Jiang, Zhendong Wang, Jianmin Bao, Wengang Zhou, Dongdong Chen, Lei Shi, Dong Chen, and Houqiang Li. Smarteraser: Remove anything from images using masked-region guidance, 2025. URL https://arxiv.org/abs/2501.08279.

Xuan Ju, Xian Liu, Xintao Wang, Yuxuan Bian, Ying Shan, and Qiang Xu. Brushnet: A plug-and-play image inpainting model with decomposed dual-branch diffusion, 2024. URL https://arxiv.org/abs/2403.06976.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Jiacheng Liu, Xinyu Wang, Yuqi Lin, Zhikai Wang, Peiru Wang, Peiliang Cai, Qinming Zhou, Zhengan Yan, Zexuan Yan, Zhengyi Shi, Chang Zou, Yue Ma, and Linfeng Zhang. A survey on cache methods in diffusion models: Toward efficient multi-modal generation, 2025a. URL https://arxiv.org/abs/2510.19755.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. *arXiv preprint arXiv:2503.06923*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Yue Ma, Zhikai Wang, Tianhao Ren, Mingzhe Zheng, Hongyu Liu, Jiayi Guo, Kunyu Feng, Yuxuan Xue, Zixiang Zhao, Konrad Schindler, Qifeng Chen, and Linfeng Zhang. Fastvmt: Eliminating redundancy in video motion transfer, 2026. URL https://arxiv.org/abs/2602.05551.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Runpu Wei, Zijin Yin, Shuo Zhang, Lanxiang Zhou, Xueyi Wang, Chao Ban, Tianwei Cao, Hao Sun, Zhongjiang He, Kongming Liang, and Zhanyu Ma. Omnieraser: Remove objects and their effects in images with paired video-frame data, 2025. URL https://arxiv.org/abs/2501.07397.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners, 2025. URL https://arxiv.org/abs/2509.20328.

Jay Zhangjie Wu, Xuanchi Ren, Tianchang Shen, Tianshi Cao, Kai He, Yifan Lu, Ruiyuan Gao, Enze Xie, Shiyi Lan, Jose M. Alvarez, Jun Gao, Sanja Fidler, Zian Wang, and Huan Ling. Chronoedit: Towards temporal reasoning for image editing and world simulation, 2025. URL https://arxiv.org/abs/2510.04290.

Yuxuan Xue, Ruofan Liang, Egor Zakharov, Timur Bagautdinov, Chen Cao, Giljoo Nam, Shunsuke Saito, Gerard Pons-Moll, and Javier Romero. Georelight: Learning joint geometrical relighting and reconstruction with flexible multi-modal diffusion transformers, 2026. URL https://arxiv.org/abs/2604.20715.

Xiangpeng Yang, Ji Xie, Yiyuan Yang, Yan Huang, Min Xu, and Qiang Wu. Unified video editing with temporal reasoner. *arXiv preprint arXiv:2512.07469*, 2025.

Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023.

Yongsheng Yu, Ziyun Zeng, Haitian Zheng, and Jiebo Luo. Omnipaint: Mastering object-oriented editing via disentangled insertion-removal inpainting. In *ICCV*, pp. 17324–17334, 2025.

Yu Zeng, Zhe Lin, and Vishal M. Patel. Sketchedit: Mask-free local image manipulation with partial sketches, 2021. URL https://arxiv.org/abs/2111.15078.

Jianhui Zhang, Sheng Cheng, Qirui Sun, Jia Liu, Wang Luyang, Chaoyu Feng, Chen Fang, Lei Lei, Jue Wang, and Shuaicheng Liu. Ultra high-resolution image inpainting with patch-based content consistency adapter, 2025. URL https://arxiv.org/abs/2510.13419.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924.

Jixin Zhao, Zhouxia Wang, Peiqing Yang, and Shangchen Zhou. Precise object and effect removal with adaptive target-aware attention. In *CVPR*, 2026.

Qinming Zhou, Chenxi Sun, Deyang Kong, Junhao He, Xiangheng Tang, Peike Yu, Haotian Wu, Leilei Cao, and Linfeng Zhang. Osor: One-step diffusion inpainting for effect-aware object removal, 2026. URL https://arxiv.org/abs/2606.28094.

APPENDIX

A Dynamic Videos **S1**

B Implementation Details **S2**

B.1 Model Configuration S2

B.2 Training Details S2

B.3 Inference Details S2

C Ablation Studies **S3**

D Benchmark Construction **S6**

D.1 Overview S6

D.2 Annotation Details S6

D.3 Evaluation Protocol S7

E Why Structured Video Rollout? **S7**

F Limitations of External Segmentation for Scribble-guided Removal **S7**

G Human-Annotated Scribble Evaluation **S8**

H Additional Experimental Results **S9**

I User Study Details **S10**

I.1 Protocol S10

I.2 Participants and Aggregation S11

I.3 Interface S12

J Limitations **S12**

J.1 Computational Cost Analysis S13

J.2 Inference Acceleration with TaylorSeer S13

K Pseudo-Code **S14**

K.1 Training Procedure S14

K.2 Scribble Mask Generation S14

K.3 Inference Procedure S14

A DYNAMIC VIDEOS

We provide a supplementary video illustrating the RolloutRemover workflow and more qualitative results, available in `demo.mp4`.

B Implementation Details

B.1 Model Configuration

Backbone. We build RolloutRemover on the Wan2.1-T2V-1.3B (Wan et al., 2025) video diffusion model, a 1.3-billion-parameter DiT architecture with 30 transformer layers, hidden dimension 1536, 12 attention heads, and patch size $(1, 2, 2)$. The model operates in a 16-channel latent space produced by a 3D causal VAE with spatial compression ratio $8\times$ and temporal compression ratio $4\times$. We use uMT5-XXL as the text encoder.

LoRA fine-tuning. We fine-tune the backbone with LoRA (rank 128) applied to the attention layers, keeping the pretrained weights frozen. This allows efficient adaptation while preserving the video prior learned during pretraining.

Sequence layout. The full input sequence has 14 frames at 480×832 pixel resolution (corresponding to 60×104 in latent space): 1 scribble-intent frame, 1 removal-region frame, 1 source-image frame, 10 rollout frames ($K_c{=}4$ corruption + 1 noise anchor + $K_r{=}5$ restoration), and 1 target frame.

Latent representation. The source image, target image, scribble mask, and removal-region mask are each encoded by the VAE encoder into a single-frame latent of shape $(16, 1, 60, 104)$. Before VAE encoding, binary masks are replicated across three channels and mapped to $[-1, 1]$; the binary mask used for the noise anchor is resized to the latent grid with nearest-neighbor interpolation. The rollout frames are constructed directly in latent space via interpolation (Section 3.2) without additional VAE calls.

B.2 Training Details

Data. We train on approximately 50k image pairs from OBER and CORNE, each containing a source image, object-removed target, object mask, and object-effect mask. Scribble masks are generated procedurally from the object masks (Algorithm 2). All keyframe latents are pre-computed and cached for efficient training.

Optimization. We use AdamW with learning rate 1×10^{-4}, weight decay 3×10^{-2}, and numerical stability constant $\epsilon{=}10^{-10}$. Training uses bf16 mixed precision with DeepSpeed ZeRO Stage-2 on 2 GPUs, per-GPU batch size 4, for a total effective batch size of 8. The maximum gradient norm is clipped to 0.05.

Training objective. We use a rectified flow-matching objective with uniform timestep sampling. Noise is added to all frames simultaneously, but condition frames (scribble-intent f_1 and source-image f_3) are restored to their clean values before the forward pass, and their corresponding positions are masked out from the loss.

Rollout hyperparameters. The corruption phase uses $K_c{=}4$ intermediate frames, the restoration phase uses $K_r{=}5$ intermediate frames, and the interpolation exponent is $\gamma{=}1.2$.

B.3 Inference Details

Inference inputs. At inference time, the model receives the source image and a condition mask. For SGORBench, this is the corresponding scribble, object, or object-effect mask; for OmniPaint-Bench, it is the provided object mask. The mask and source image are encoded as clean frames f_1 and f_3, respectively. All other frames are initialized with Gaussian noise and generated jointly.

Sampling. We use 50 denoising steps with classifier-free guidance (scale 5.0), the fixed positive prompt "Remove the masked object and its side effect", and an empty negative prompt. At each step, condition frames are restored after the scheduler update to maintain clean conditioning throughout generation.

Decoding. Although the Wan2.1 backbone uses a 3D causal VAE, we decode each latent frame independently to obtain per-frame output images. The final object-removed result is taken from the last frame f_N of the decoded sequence.

C Ablation Studies

We conduct ablation studies on the 1.3B-parameter model to analyze the key design choices of RolloutRemover. Unless otherwise noted, all variants share the same training data, training steps, and backbone architecture; only the specified component is modified. We report PSNR, SSIM, LPIPS, FID, CFD, and ReMOVE on the scribble mask evaluation set (300 samples).

Rollout construction method. We first compare our two-phase corruption-restoration rollout against alternative rollout construction strategies that use the same number of intermediate frames but differ in how those frames are generated.

Table 1: Ablation on rollout construction method. All variants use the same total number of intermediate frames ($K_c+K_r+1 = 10$). "Ours" refers to the corruption-restoration rollout; "Linear" and "Slerp" replace the rollout with direct source-to-target interpolation.

Rollout Construction	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
Ours (corruption-restoration)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
Source-target slerp	<u>26.42</u>	<u>0.8553</u>	0.1499	<u>41.82</u>	<u>0.2782</u>	0.9199
Source-target linear	25.46	0.8525	<u>0.1461</u>	44.40	0.3176	<u>0.9223</u>

Table 1 shows that directly interpolating between the source and target latents, either through linear interpolation or spherical linear interpolation (slerp), consistently underperforms our corruption-restoration construction on most evaluation metrics. Compared with slerp, our rollout improves PSNR from 26.42 to 27.00 dB, reduces LPIPS from 0.1499 to 0.1368, and lowers FID from 41.82 to 36.37. The advantage is even larger over linear interpolation, with a 1.54 dB improvement in PSNR and an 8.03-point reduction in FID. These results suggest that explicitly introducing a noise anchor and decomposing the transformation into object corruption followed by background restoration provides more effective process supervision than directly interpolating between the source and target states.

Rollout ordering. We next examine whether the ordering of intermediate states contributes to the effectiveness of the rollout, rather than the performance gain arising solely from the additional supervision frames.

Table 2: Ablation on rollout frame ordering. "Default" is the standard corruption-then-restoration order. "Reversed" reverses the inner frames. "Shuffled" applies a fixed random permutation to the inner frames.

Ordering	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
Default	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
Reversed	<u>26.74</u>	<u>0.8595</u>	<u>0.1426</u>	<u>39.23</u>	<u>0.2721</u>	<u>0.9237</u>
Shuffled	25.66	0.8537	0.1576	56.52	0.3933	0.9048

Table 2 compares the default corruption-then-restoration order with reversed and shuffled variants while keeping the same set of intermediate frames. Reversing the inner sequence causes a moderate degradation, reducing PSNR from 27.00 to 26.74 dB and increasing FID from 36.37 to 39.23. Randomly shuffling the intermediate states leads to a substantially larger degradation, with PSNR dropping to 25.66 dB, FID increasing to 56.52, and ReMOVE decreasing from 0.9334 to 0.9048. The progressively larger degradation from the default to reversed and shuffled settings indicates that the temporal organization of intermediate states is important, rather than the benefit coming simply from providing additional frames. In particular, maintaining a coherent corruption-then-restoration progression provides the most effective form of process supervision.

Removal-region frame supervision. We further investigate the contribution of the explicit removal-region frame f_2, which provides intermediate supervision for inferring the complete removal extent from sparse scribbles. We compare the full model with a variant that removes the middle removal-region frame and directly connects the scribble-intent frame to the rollout process.

Table 3: Ablation on the removal-region frame f_2. "w/o f_2" removes the intermediate removal-region frame and directly performs rollout from the scribble condition.

Variant	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
w. f_2	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
w/o f_2	25.87	0.8510	0.1561	44.05	0.3012	0.9212

Table 3 shows that removing the intermediate removal-region frame consistently degrades perceptual and reconstruction metrics. Without f_2, the model needs to infer the complete removal extent and perform the subsequent rollout simultaneously, which increases the difficulty of scribble interpretation. In contrast, explicitly supervising the removal region provides an intermediate semantic representation between sparse user intent and the removal trajectory, leading to improved object removal quality.

Interpolation schedule. We vary the interpolation exponent γ that controls the curvature of both the corruption and restoration schedules.

Table 4: Ablation on the interpolation exponent γ used in rollout construction.

γ	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
1.0	<u>25.66</u>	<u>0.8536</u>	<u>0.1500</u>	<u>50.71</u>	<u>0.3475</u>	<u>0.8870</u>
1.2 (ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
1.5	25.40	0.8529	0.1556	58.79	0.3749	0.8825

Table 4 shows that the interpolation schedule has a noticeable impact on performance. Using uniform linear spacing ($\gamma = 1.0$) reduces PSNR from 27.00 to 25.66 dB and increases FID from 36.37 to 50.71. Increasing the exponent further to $\gamma = 1.5$ leads to an even larger degradation, with FID increasing to 58.79 and ReMOVE decreasing to 0.8825. In contrast, $\gamma = 1.2$ consistently achieves the best performance across all six metrics. This suggests that a mild nonlinear schedule provides a more effective allocation of intermediate states than either uniform spacing or a more strongly curved trajectory. We therefore adopt $\gamma = 1.2$ in the final model.

Rollout supervision versus endpoint prediction. To verify that the performance gain comes from the explicit rollout rather than simply adopting a video backbone, we compare the full model with an endpoint-only variant that keeps the same conditioning frames but removes all intermediate rollout frames, directly predicting the target from the conditioning frames.

Table 5: Ablation on rollout supervision versus endpoint prediction. "Full rollout" uses the complete corruption-restoration sequence; "Endpoint only" removes all intermediate rollout frames and directly predicts the target from the conditioning frames.

Variant	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
Full rollout (ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
Endpoint only	25.79	0.8529	0.1467	50.40	0.3556	0.9170

Table 5 confirms a clear advantage of explicit rollout supervision over endpoint-only prediction. Although both variants use the same video diffusion backbone and conditioning setup, removing the intermediate rollout frames decreases PSNR by 1.21 dB and increases FID from 36.37 to 50.40, corresponding to a 38.6% relative increase. CFD also increases from 0.2670 to 0.3556, a 33.1% relative increase, while ReMOVE decreases from 0.9334 to 0.9170. These results indicate that the improvement is not simply attributable to using a video diffusion backbone; instead, the intermediate states provide additional structured supervision that substantially benefits object removal quality.

Necessity of corruption and restoration phases. We further ablate the two-phase design by removing each phase independently.

Table 6: Ablation on the two-phase rollout design. "No corruption" removes the corruption phase (K_c=0); "No restoration" removes the restoration phase (K_r=0).

Variant	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
Full model (ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
No corruption (K_c=0)	24.59	0.8342	0.1777	62.68	0.3760	0.9046
No restoration (K_r=0)	25.00	0.8484	0.1600	52.96	0.3321	0.9206

Table 6 demonstrates that both phases are important for the overall performance. Removing the corruption phase reduces PSNR by 2.41 dB and increases FID from 36.37 to 62.68, while removing the restoration phase reduces PSNR by 2.00 dB and increases FID to 52.96. The absence of either phase also consistently worsens LPIPS, CFD, and ReMOVE. In particular, removing the corruption phase causes a larger degradation, suggesting that explicit supervision for the object-disappearance process is especially important. At the same time, the degradation caused by removing the restoration phase confirms that explicitly modeling background recovery is also necessary. Together, these results support the complementary roles of the two phases in the proposed rollout.

Corruption phase length. We vary the number of corruption frames while holding the restoration phase fixed at K_r=5.

Table 7: Ablation on corruption phase length K_c with fixed restoration length K_r=5.

K_c	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
0	24.59	0.8342	0.1777	62.68	0.3760	0.9046
2	26.05	0.8511	0.1555	47.23	0.3191	0.9111
4 (ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
6	23.92	0.8416	0.1768	69.28	0.4585	0.7895

Table 7 shows that increasing the number of corruption frames from $K_c = 0$ to $K_c = 4$ consistently improves performance. The best results are obtained at $K_c = 4$, which achieves 27.00 dB PSNR and 36.37 FID. However, further increasing the corruption phase to $K_c = 6$ causes a substantial degradation, with PSNR dropping to 23.92 dB and FID increasing to 69.28. This indicates that an excessively long corruption phase may over-emphasize the object-disappearance process and make the learned transition trajectory less effective for subsequent background recovery. Thus, $K_c = 4$ provides a favorable balance between explicitly modeling object disappearance and maintaining effective restoration dynamics.

Restoration phase length. We vary the restoration phase length while holding the corruption phase fixed at K_c=4.

Table 8: Ablation on restoration phase length K_r with fixed corruption length K_c=4.

K_r	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
0	25.00	0.8484	0.1600	52.96	0.3321	0.9206
3	25.59	0.8534	0.1550	52.25	0.3652	0.9047
5 (ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
7	25.96	0.8526	0.1541	42.78	0.3202	0.9015

Table 8 shows that introducing restoration frames provides a substantial benefit, with the best overall performance achieved at $K_r = 5$. Compared with $K_r = 0$, increasing the restoration length to 5 improves PSNR from 25.00 to 27.00 dB and reduces FID from 52.96 to 36.37. However, extending the restoration phase to $K_r = 7$ leads to a degradation across most metrics, with PSNR decreasing to 25.96 dB and FID increasing to 42.78. This indicates that while additional restoration states improve supervision up to a certain point, an excessively long restoration phase can make the rollout less balanced and reduce the effectiveness of the learned trajectory. We therefore adopt $K_r = 5$ as the final configuration.

D BENCHMARK CONSTRUCTION

D.1 OVERVIEW

We construct SGORBench from publicly available object removal datasets, specifically OBER and RORD-Val (Zhao et al., 2026) and CORNE (Zhou et al., 2026). The benchmark contains 300 aligned samples that are isolated from the training data at both the case and similar-scene levels. Each sample is evaluated under three mask conditions:

- **Scribble mask condition**: the condition mask is a procedurally generated scribble that simulates casual user interaction. The scribble is not required to fully cover the target object and may extend slightly beyond its boundary.
- **Object mask condition**: the condition mask is the object segmentation mask marking the spatial extent of the target object.
- **Object-effect mask condition**: the condition mask is the object-effect mask covering both the target object and its associated effects (e.g., shadows, reflections, contact traces). Samples in this set are specifically selected from cases where the object-effect mask differs from the object mask, ensuring the presence of visible object-associated effects.

The three mask conditions are evaluated on the same 300 samples, enabling controlled comparison across input mask types. Figure 1 presents examples from SGORBench, illustrating the three mask types across diverse scenes and target objects.

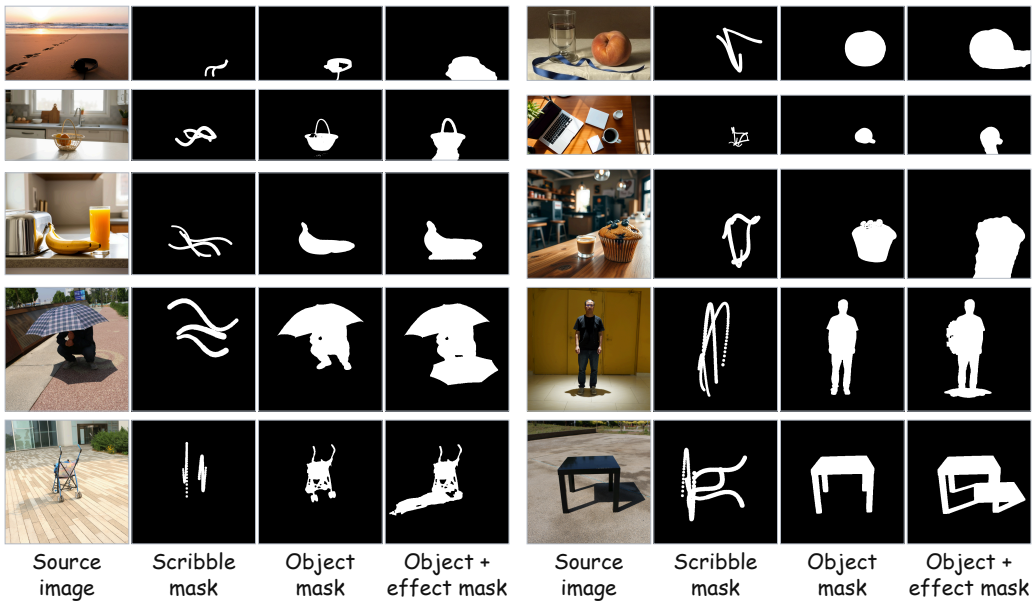


Figure 1: **SGORBench example.** Each group shows the source image, scribble mask, object mask, and object-effect mask. The object mask delineates the target object, whereas the object-effect mask additionally includes associated effects. White indicates the masked region, and black indicates the background.

D.2 ANNOTATION DETAILS

Condition masks. Each sample provides a condition mask that serves as the input to all evaluated methods. For the scribble set, we generate scribble masks from the ground-truth object masks using the procedural generation described in Algorithm 2. For the object mask and object-effect mask sets, the condition masks are taken directly from the dataset annotations.

Ground truth. Each sample provides a ground-truth object-removed target image, along with the original source image.

D.3 Evaluation Protocol

All methods receive the source image and the condition mask as input and produce a single object-removed output image. We evaluate using six metrics: PSNR, SSIM, LPIPS, FID, CFD, and ReMOVE. This protocol tests not only removal quality but also robustness to the form of user input: the scribble set evaluates whether a method can infer removal intent from sparse indication, the object mask set evaluates standard removal quality, and the object-effect mask set evaluates the ability to handle associated effects when explicitly specified.

E Why Structured Video Rollout?

Purpose. RolloutRemover differs from existing image-based removers by casting object removal as structured generation over a video rollout. We therefore ask whether adapting image-based removers to our training data can close the performance gap with RolloutRemover. To investigate this question, we fine-tune two representative image-based removers, OmniEraser (Wei et al., 2025) and SmartEraser (Jiang et al., 2025), while retaining their original image-based formulations. We compare their released and fine-tuned variants with RolloutRemover on the SGORBench scribble set using the six reported metrics.

Data and fine-tuning setup. Starting from each method's released checkpoint, we fine-tune both baselines on the same training data used to train RolloutRemover. Each baseline is trained for 2400 steps at an effective batch size of 6. OmniEraser uses rank-32 LoRA adaptation with 8-bit AdamW, an effective learning rate of 1.8×10^{-4}, and its native latent flow-matching MSE objective. For SmartEraser, we fine-tune the full UNet, CLIP text encoder, and visual-prompt MLP using AdamW, an effective learning rate of 3×10^{-6}, and its native latent diffusion denoising MSE objective; the VAE and CLIP vision encoder remain frozen.

Table 9: Comparison of RolloutRemover with released and fine-tuned image-based removers on SGORBench under the **scribble mask** condition. Best results are in **bold**, and second-best results are underlined. All predictions are evaluated in the corresponding source-image geometry.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
OmniEraser (released)	21.92	0.7530	0.2019	71.26	0.3868	0.8149
OmniEraser (fine-tuned)	<u>24.14</u>	<u>0.8145</u>	<u>0.1577</u>	<u>52.80</u>	0.3099	0.8973
SmartEraser (released)	21.87	0.7657	0.1891	77.36	0.4432	0.7459
SmartEraser (fine-tuned)	19.90	0.7204	0.2277	64.11	<u>0.2860</u>	<u>0.9134</u>
RolloutRemover (Ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**

Results. Table 9 shows that fine-tuning substantially improves OmniEraser across all six metrics, whereas SmartEraser improves in FID, CFD, and ReMOVE but declines in PSNR, SSIM, and LPIPS. Nevertheless, both fine-tuned image-based removers remain behind RolloutRemover on every metric, despite adaptation with matched training samples and sample-exposure budgets. Rather than predicting only the final removed image, RolloutRemover represents removal as a structured latent rollout that explicitly separates object corruption from background restoration. These results support our central premise that structured video-rollout modeling provides a more effective formulation for scribble-guided object removal than direct image-based approaches evaluated here.

F Limitations of External Segmentation for Scribble-guided Removal

Motivation. A possible concern is that mask-conditioned baselines have lower performance because sparse scribbles do not match their expected dense-mask inputs. We therefore test whether an external segmentation model can bridge this interface gap.

Setup. We use a frozen SAM 2.1 model (Ravi et al., 2025) to convert user scribbles into dense condition masks. We rerun all five mask-conditioned baselines with the same SAM-derived masks while

retaining each method's frozen checkpoint and inference settings. By contrast, RolloutRemover natively accepts scribble guidance and is evaluated directly using the original scribble, without external segmentation. All outputs are restored to source geometry and evaluated using the same evaluator on our SGORBench set.

Table 10: Quantitative comparison on the SGORBench set. The five baselines use masks generated by the same frozen SAM 2.1 model, whereas RolloutRemover directly uses the original scribble. Best results are in **bold**, and second-best results are underlined.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
ObjectClear + SAM	19.12	0.7014	0.2788	87.71	0.4316	0.8002
OmniPaint + SAM	17.90	0.6916	0.3085	105.53	0.4567	0.7369
OSOR + SAM	<u>22.94</u>	<u>0.8210</u>	<u>0.1801</u>	<u>80.17</u>	0.4274	0.7927
SmartEraser + SAM	19.08	0.6881	0.2968	86.00	<u>0.3785</u>	0.8057
OmniEraser + SAM	19.80	0.6789	0.2842	85.15	0.3972	<u>0.8290</u>
RolloutRemover (Ours)	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**

Results. As shown in Table 10, RolloutRemover achieves the best performance across all six metrics. Importantly, none of the five mask-conditioned baselines matches RolloutRemover even after the sparse scribble is converted into a SAM-derived condition mask. This result suggests that the gap in the original comparison is not solely explained by the baselines treating the scribble itself as the removal region, thereby addressing a key concern about baseline fairness. RolloutRemover therefore provides a more effective solution for interpreting sparse scribble intent and achieving complete object removal.

G Human-Annotated Scribble Evaluation

Our benchmark uses procedurally generated scribbles to enable controlled and scalable evaluation. To examine whether these synthetic scribbles capture the characteristics of natural human guidance, we first qualitatively compare procedurally generated scribbles with human-annotated scribbles on the same object-removal samples.

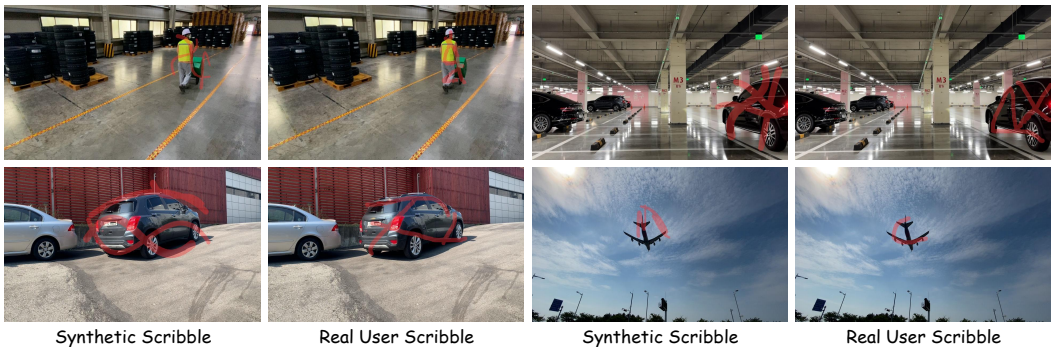


Figure 2: Qualitative comparison between procedurally generated scribbles and human-annotated scribbles on the same object-removal samples. Both types of scribbles exhibit natural variations in stroke patterns, object coverage, and boundary precision.

As shown in Figure 2, human-annotated scribbles exhibit substantial variations in stroke patterns, object coverage, and boundary precision. Our procedurally generated scribbles capture these characteristics, including sparse indications, incomplete object coverage, irregular strokes, and occasional imprecise boundaries. This qualitative comparison suggests that the procedural generation process provides a reasonable approximation of casual human guidance while retaining the controllability required for large-scale benchmark construction.

To further evaluate whether models trained and evaluated under synthetic scribbles generalize to human-provided guidance, we conduct a quantitative evaluation using human-annotated scribbles.

Specifically, we manually annotate scribbles for all 300 samples in SGORBench, replacing the procedurally generated scribbles with human annotations while keeping the source images, ground-truth masks, and evaluation protocol unchanged. All compared methods are then evaluated using these human-annotated scribbles under the same experimental settings as the main benchmark.

Table 11: Quantitative comparison on 300 samples with human-annotated scribbles. All methods receive the same human-annotated scribble inputs. The best and second-best results are shown in **bold** and underlined, respectively.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
RolloutRemover	26.9852	**0.8615**	**0.1363**	**36.05**	**0.2987**	**0.9331**
OSOR	**26.9893**	0.8356	0.1492	49.95	0.3044	0.9046
ObjectClear	23.4487	0.7875	0.1475	65.42	0.4233	0.8186
SmartEraser	22.7781	0.7713	0.1763	65.27	0.3904	0.7898
OmniEraser	22.1067	0.7586	0.1994	67.77	0.3834	0.8310
OmniPaint	20.9817	0.8496	0.1373	102.98	0.5639	0.6258

Table 11 shows that RolloutRemover maintains strong performance when the synthetic scribbles are replaced by human-annotated scribbles. It achieves the best results on five of the six evaluation metrics, including SSIM, LPIPS, FID, ReMOVE, and CFD. In particular, RolloutRemover achieves an FID of 36.0475, substantially lower than the second-best result of 49.9488 from OSOR, corresponding to a 27.8% relative reduction. It also improves ReMOVE from 0.9046 to 0.9331 and reduces LPIPS from 0.1492 to 0.1363, indicating improved removal quality and perceptual fidelity under human-provided guidance. The CFD score is similarly reduced from 0.3044 to 0.2987.

For PSNR, OSOR obtains a marginally higher score of 26.9893 dB, compared with 26.9852 dB for RolloutRemover. The difference is only 0.0041 dB, while RolloutRemover substantially outperforms OSOR on the other perceptual and removal-oriented metrics. This suggests that the advantage of RolloutRemover under human annotations is not merely reflected by pixel-level reconstruction fidelity, but is more pronounced in perceptual quality and removal consistency.

Overall, these results demonstrate that RolloutRemover generalizes well from procedurally generated scribbles to human-annotated scribbles. Despite the distribution shift in scribble patterns and coverage, RolloutRemover consistently achieves superior performance across perceptual and removal-oriented metrics. Together with the qualitative comparison in Figure 2, these results indicate that our procedural scribble construction provides a meaningful approximation of human guidance, while the proposed model does not rely on the specific geometric patterns of the synthetic scribbles.

H Additional Experimental Results

Visual comparisons. We include more visual examples comparing RolloutRemover with prior methods under standard and imperfect scribble inputs, as shown in Figure 3.

Qualitative gallery. We provide additional examples of RolloutRemover across diverse scenarios, including large target areas, structured backgrounds and object-associated effects, as shown in Figure 4.

Failure cases. We further present several cases where RolloutRemover does not fully remove the intended target, as shown in Figure 5. A consistent pattern across these cases is that the input scribble is substantially misaligned with the target or covers only a small part of it. Such guidance does not clearly indicate the full target to be removed. Consequently, RolloutRemover follows the region indicated by the scribble rather than the full object, leaving unmarked target components partially or entirely visible.

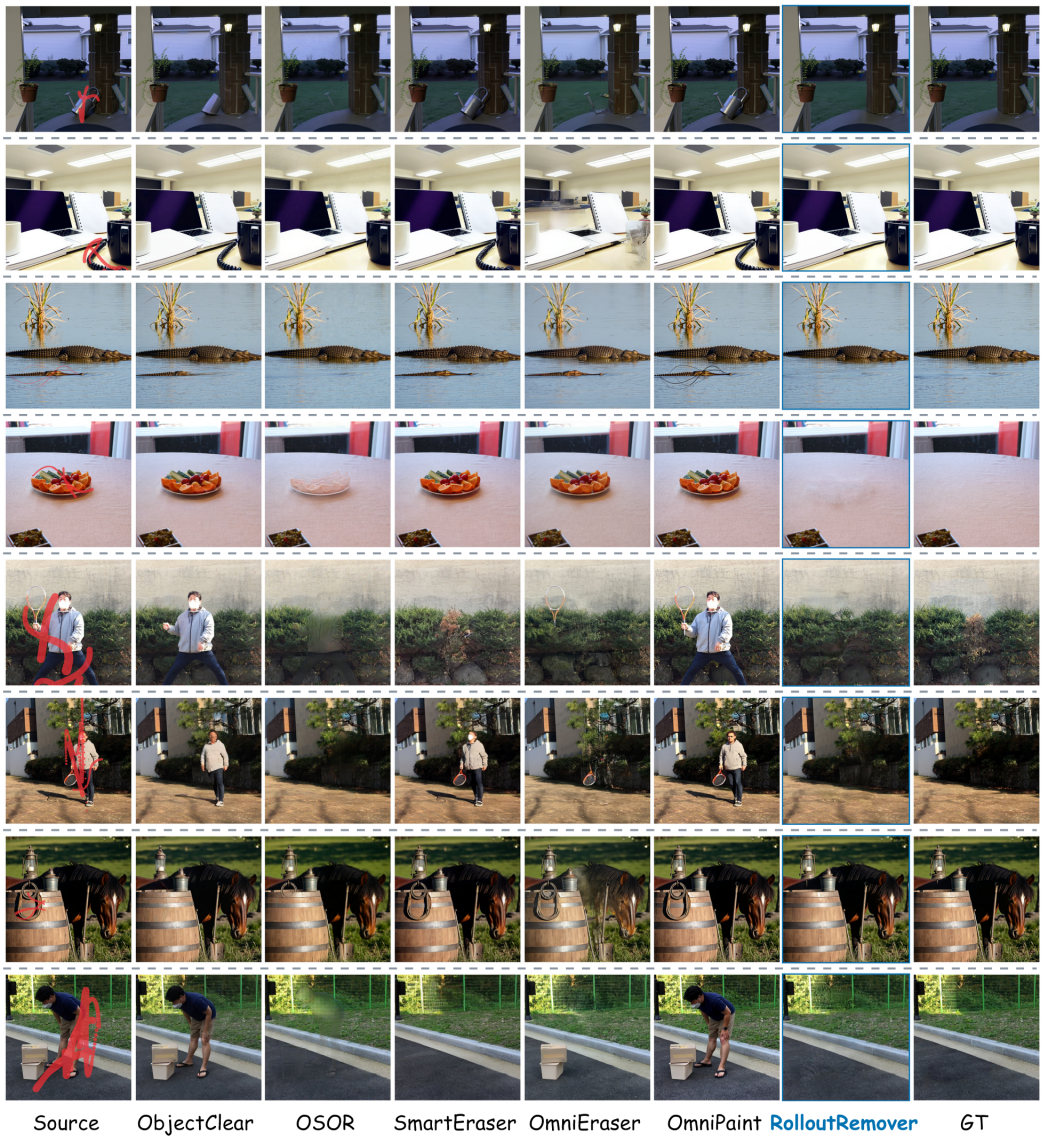


Figure 3: More qualitative comparisons of RolloutRemover and prior methods under scribble guidance.

I User Study Details

I.1 Protocol

Presentation format. Participants are shown the source image with the scribble mask and six candidate result images. The corresponding method identities are hidden from participants.

Randomization. For each participant and case, the mapping between displayed labels and methods is randomized independently to mitigate presentation bias.

Evaluation criteria. For each case, participants select the Top-2 candidates separately under each of the following five criteria:

- **Removal completeness:** whether the target object and its associated effects are fully removed with minimal visible remnants.

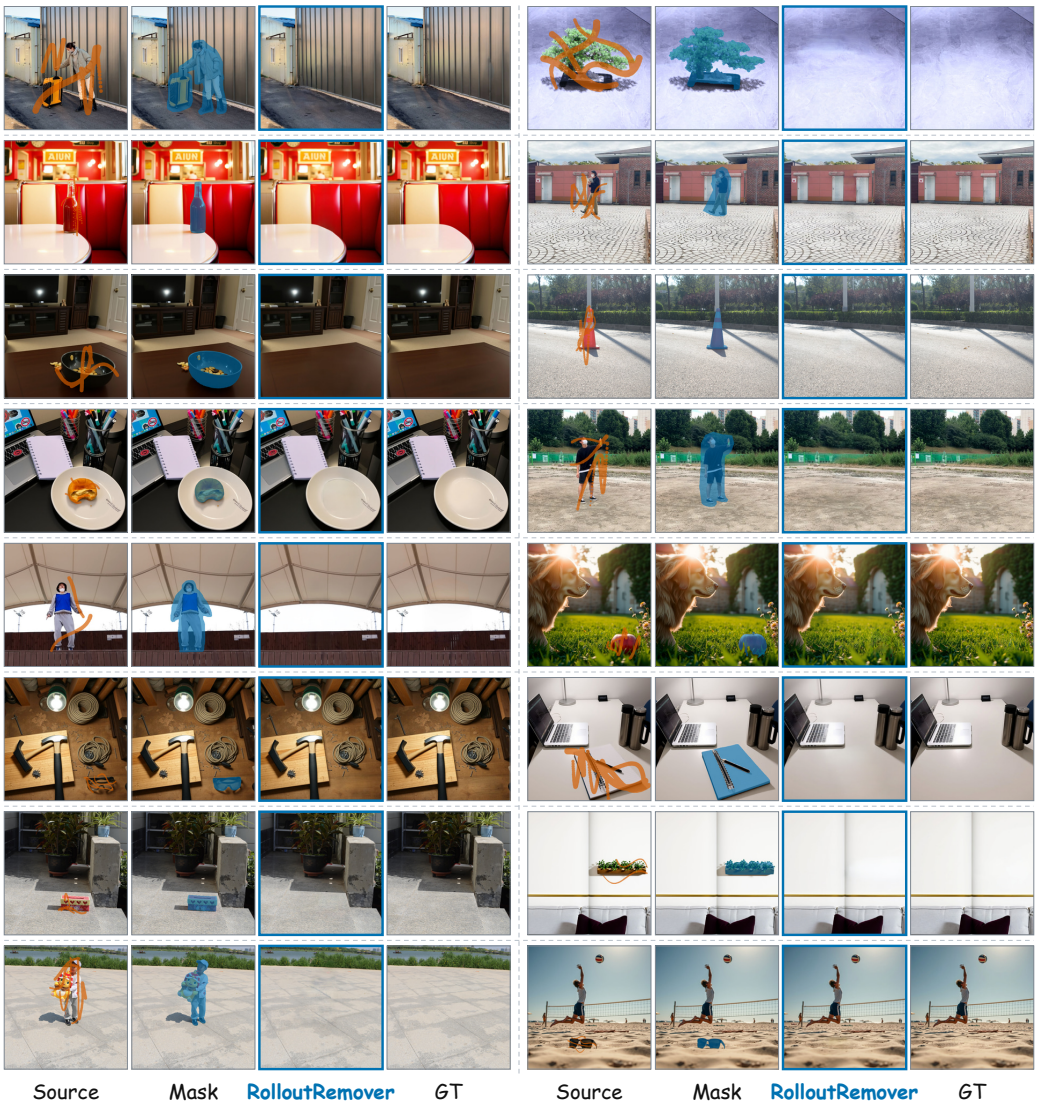


Figure 4: More qualitative results of RolloutRemover across diverse object-removal scenarios.

- **Background plausibility:** whether the background restored in the removed region is consistent with the surrounding scene and appears realistic and natural.
- **Edge seamlessness:** whether the boundary of the removed region blends naturally with its surroundings without visible seams or artifacts.
- **Non-target preservation:** whether non-target objects, scene content, and details are preserved well without unintended removals or unrelated changes.
- **Overall performance:** whether the result successfully accomplishes the object-removal task and is suitable as the final output.

We treat Overall Performance as the primary criterion.

I.2 Participants and Aggregation

Participants. The user study used two forms, each containing 10 object removal cases. A total of 60 participants completed the user study. Each form was evaluated by a separate group of 30



Figure 5: Failure cases of RolloutRemover under misaligned or incomplete scribble guidance.

participants. Each participant evaluated all 10 cases in one form. For each case and criterion, the participant made two unordered selections from the six candidate results. All participants were student volunteers and received no compensation. None was an author of this paper.

Aggregation. In our user study, the two Top-2 selections are treated as unordered, and each selected candidate receives one vote. We aggregate these votes by method and report the Top-2 selection rate, which indicates the selected frequency of each method. The selection rate is computed separately for the two forms and averaged equally. For the stacked-bar visualization in Figure 5(a), the votes are normalized so that the method shares sum to 100% for each evaluation criterion.

I.3 Interface

User study page. Figures 6 and 7 show the two components of the user study interface. Participants viewed a scribble-annotated input and six method-blinded candidate outputs, then selected Top-2 candidates under each of five evaluation criteria.

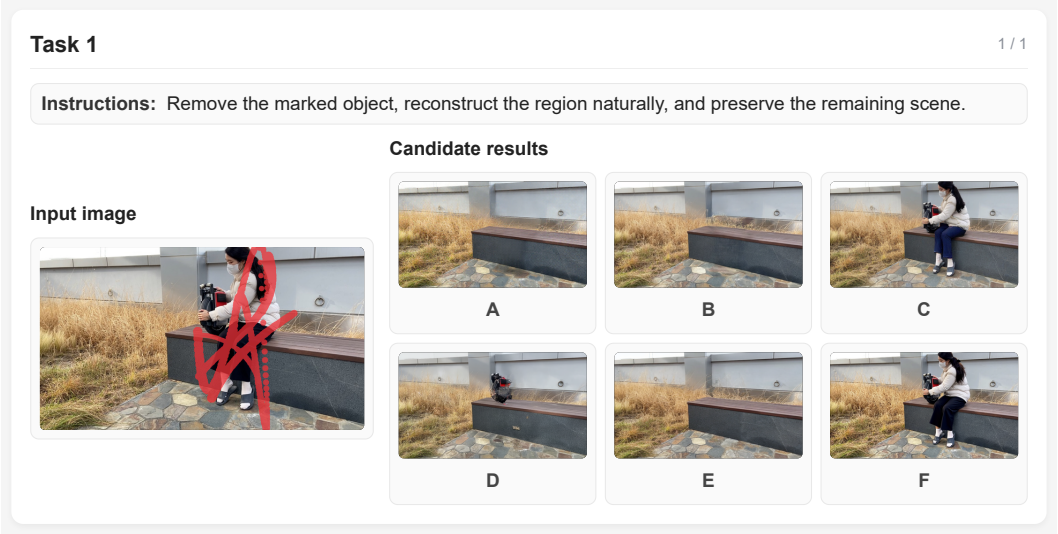


Figure 6: Task presentation and candidate outputs in the user study interface.

J Limitations

Although RolloutRemover is more robust to sparse scribbles than previous mask-based pipelines, it may still struggle when user intent is ambiguous, such as when multiple nearby objects could be potential targets. In addition, the video diffusion backbone introduces non-negligible inference

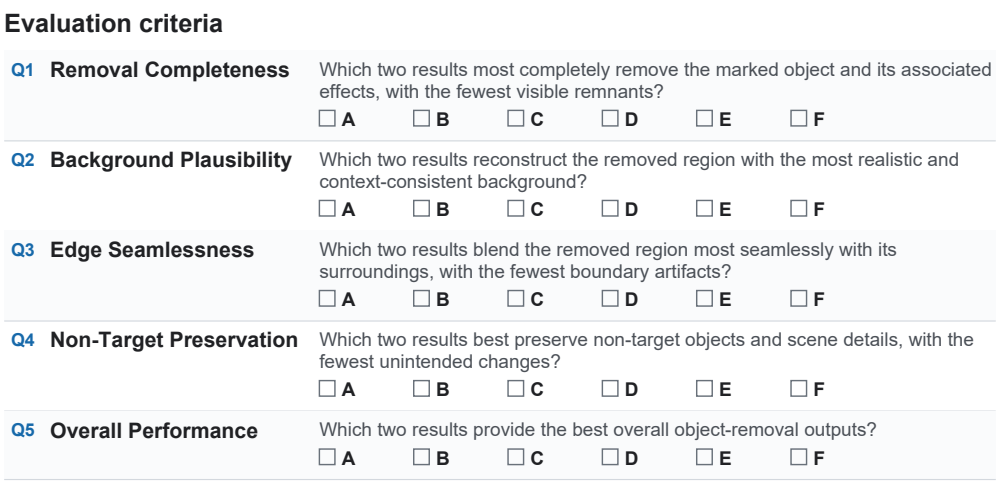


Figure 7: Evaluation criteria and Top-2 selection controls in the user study interface.

latency. However, this computational limitation can be substantially alleviated by inference acceleration techniques.

J.1 Computational Cost Analysis

To characterize the practical computational requirements of our method, we compare the inference cost of our approach with representative object removal baselines in terms of parameter count, inference steps, latency, and peak GPU memory consumption. The results are summarized in Table 12. All measurements are reported under the inference settings used in our experiments.

Table 12: Computational cost comparison with representative object removal methods. Params denotes the total number of model parameters in millions. Latency and VRAM denote the measured inference time and peak GPU memory consumption, respectively.

Method	Backbone	Params (M)	Steps	Latency (s)	VRAM (GB)
OSOR	SDXL UNet + LoRA	3,345	1	0.29	7.4
SmartEraser	SD UNet	860	50	4.05	7.7
ObjectClear	SDXL UNet	2,567	20	3.11	8.1
OmniPaint	FLUX.1-dev + LoRA	11,916	28	14.37	32.5
OmniEraser	FLUX.1-dev + LoRA	11,992	28	15.13	34.0
RolloutRemover (Ours)	Wan2.1-1.3B DiT + LoRA	1,419	50	78.83	17.53

Compared with image-based diffusion models, our method requires longer inference time due to the multi-frame generation process. Nevertheless, this latency bottleneck can be substantially alleviated by inference acceleration, with only minimal degradation in removal quality.

J.2 Inference Acceleration with TaylorSeer

Inspired by prior work on caching for diffusion models (Liu et al., 2025a), we apply TaylorSeer (Liu et al., 2025b) to reduce redundant computation during denoising while maintaining comparable removal quality. Specifically, we apply TaylorSeer with $cache_step = 6$ and $first_enhance = 2$.

To examine the effect of acceleration on output quality, we compare TaylorSeer with our model without acceleration. As shown in Table 13, TaylorSeer introduces only minimal degradation in removal quality while reducing the inference latency from 78.83 seconds to 22.78 seconds. Furthermore, despite the acceleration, the TaylorSeer variant remains highly competitive with existing object removal methods under the same scribble-mask setting, achieving better performance than the best baseline result on five out of six evaluation metrics.

Table 13: Quality comparison between runs with and without TaylorSeer acceleration. TaylorSeer uses $cache_step = 6$ and $first_enhance = 2$.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓	CFD↓	ReMOVE↑
w/o TaylorSeer	**27.00**	**0.8606**	**0.1368**	**36.37**	**0.2670**	**0.9334**
w/ TaylorSeer	26.60	0.859	0.143	40.26	0.2754	0.9288

These results demonstrate that TaylorSeer effectively alleviates the inference latency bottleneck of RolloutRemover while preserving its strong removal capability. With a $3.46\times$ speedup and only minor quality variation, inference caching methods provide an effective solution to improve the inference efficiency of our video-based formulation.

K Pseudo-Code

K.1 Training Procedure

Algorithm 1 describes RolloutRemover training with a scribble mask, including rollout construction and flow-matching optimization. When an object or object-effect mask is sampled, it is used in place of m_s and the remaining steps are unchanged.

K.2 Scribble Mask Generation

Algorithm 2 describes the procedural scribble generation used to construct training scribble masks from ground-truth object masks. Each scribble consists of multiple curved strokes with variable width, designed to approximate casual user interaction.

K.3 Inference Procedure

Algorithm 3 describes the inference procedure with a scribble mask. In the dense-mask evaluations, the supplied mask is used in place of m_s as the clean f_1 condition.

Algorithm 1 RolloutRemover Training

Require: Source image x_s, target image x_t, scribble mask m_s, removal-region mask m_r
Require: VAE encoder E, video diffusion model θ, corruption length K_c, restoration length K_r, exponent γ

// Encode inputs to latent space
$z_s \leftarrow E(x_s), \quad z_t \leftarrow E(x_t), \quad z_{\text{scr}} \leftarrow E(m_s), \quad z_{\text{reg}} \leftarrow E(m_r)$
$m_r^{\text{lat}} \leftarrow \texttt{Resize}(m_r, \text{latent resolution})$ *// downsample mask to latent grid*

// Construct noise anchor
$\epsilon \sim \mathcal{N}(0, I)$ with the same shape as z_s
$z_a \leftarrow (1 - m_r^{\text{lat}}) \odot z_s + m_r^{\text{lat}} \odot \epsilon$

// Build corruption phase: source → noise anchor
for $i = 1$ **to** K_c **do**
  $\alpha_i \leftarrow \left(\frac{i}{K_c+1}\right)^{\gamma}$
  $z_i^{\text{cor}} \leftarrow (1 - \alpha_i)\, z_s + \alpha_i\, z_a$
end for

// Build restoration phase: noise anchor → target
for $j = 1$ **to** K_r **do**
  $\beta_j \leftarrow 1 - \left(1 - \frac{j}{K_r+1}\right)^{\gamma}$
  $z_j^{\text{res}} \leftarrow (1 - \beta_j)\, z_a + \beta_j\, z_t$
end for

// Assemble full sequence
$\mathcal{S} \leftarrow [z_{\text{scr}},\ z_{\text{reg}},\ z_s,\ z_1^{\text{cor}}, \ldots, z_{K_c}^{\text{cor}},\ z_a,\ z_1^{\text{res}}, \ldots, z_{K_r}^{\text{res}},\ z_t]$

// Flow-matching noise addition
Sample timestep $t \sim \mathcal{U}(0, 1), \quad \epsilon' \sim \mathcal{N}(0, I)$ with the same shape as $\mathcal{S}$
$\sigma \leftarrow \sigma(t)$ *// noise schedule*
$\mathcal{S}_{\text{noisy}} \leftarrow (1 - \sigma)\, \mathcal{S} + \sigma\, \epsilon'$

// Keep condition frames clean (scribble f_1 and source f_3)
$\mathcal{S}_{\text{noisy}}[1] \leftarrow z_{\text{scr}}, \quad \mathcal{S}_{\text{noisy}}[3] \leftarrow z_s$

// Predict and compute loss (exclude condition frames)
$v_\theta \leftarrow \theta(\mathcal{S}_{\text{noisy}}, t)$ *// model prediction*
$v_{\text{target}} \leftarrow \epsilon' - \mathcal{S}$ *// flow-matching target*
$\mathcal{L} \leftarrow \|(v_\theta - v_{\text{target}}) \odot \mathbf{M}_{\text{loss}}\|^2$ *// $\mathbf{M}_{loss}$ masks out condition frames*
Update θ by $\nabla_\theta \mathcal{L}$

Algorithm 2 Scribble Mask Generation from Object Mask

Require: Binary object mask M, number of strokes $N_s \sim \mathcal{U}\{2, 5\}$
Ensure: Scribble mask S (same resolution as M)

$S \leftarrow \mathbf{0}$ *// blank canvas*
$\mathcal{C} \leftarrow \texttt{ExtractContour}(M)$ *// largest external contour*
$\mathbf{c} \leftarrow \texttt{Centroid}(M), \quad L \leftarrow \texttt{MajorAxis}(M), \quad l \leftarrow \texttt{MinorAxis}(M)$

for $n = 1$ **to** N_s **do**
  // Sample two anchor points on the contour with sufficient separation
  repeat
    $A, B \leftarrow \texttt{SampleContourPoint}(\mathcal{C})$
  until $\|A - B\| > 0.4 \cdot L$

  // Apply mild endpoint perturbation to simulate natural user strokes
  $\epsilon_A, \epsilon_B \sim \texttt{Bernoulli}(0.5)$
  $A' \leftarrow A + \epsilon_A \cdot \texttt{Normalize}(A - \mathbf{c}) \cdot \mathcal{U}(-0.05, 0.05) \cdot l$
  $B' \leftarrow B + \epsilon_B \cdot \texttt{Normalize}(B - \mathbf{c}) \cdot \mathcal{U}(-0.05, 0.05) \cdot l$

  // Construct cubic Bézier control points with random perturbation
  $\mathbf{d} \leftarrow \texttt{Normalize}(B' - A'), \quad \mathbf{p} \leftarrow \texttt{Perpendicular}(\mathbf{d})$
  $P_1 \leftarrow \frac{1}{3}A' + \frac{2}{3}B' + \mathbf{p} \cdot \mathcal{U}(-0.2, 0.2) \cdot \|B' - A'\|$
  $P_2 \leftarrow \frac{2}{3}A' + \frac{1}{3}B' + \mathbf{p} \cdot \mathcal{U}(-0.2, 0.2) \cdot \|B' - A'\|$

  // Fit smooth curve and draw with variable width
  $\texttt{curve} \leftarrow \texttt{SplineFit}([A', P_1, P_2, B'])$
  $w_{\text{start}}, w_{\text{end}} \sim \mathcal{U}(0.05, 0.12) \cdot l$
  $\texttt{DrawVariableWidthCurve}(S, \texttt{curve}, w_{\text{start}}, w_{\text{end}})$
end for

return S

Algorithm 3 RolloutRemover Inference

Require: Source image x_s, scribble mask m_s
Require: VAE encoder E, VAE decoder D, video diffusion model θ
Require: Number of denoising steps T, guidance scale w, total frames N

// Encode conditions to latent space
$z_{\text{scr}} \leftarrow E(m_s), \quad z_s \leftarrow E(x_s)$

// Initialize latent sequence with noise
$z_{\text{mid}} \sim \mathcal{N}(0, I)$ with the same shape as z_s *// for removal-region frame*
$z_{\text{tail}} \sim \mathcal{N}(0, I)$ with shape $(C, N-3, H, W)$ *// for rollout + target*
$\mathcal{S} \leftarrow [z_{\text{scr}},\ z_{\text{mid}},\ z_s,\ z_{\text{tail}}]$

// Iterative denoising
for $k = 1$ **to** T **do**
  // CFG: c_{pos}/c_{neg} denote positive/negative prompt embeddings
  $v_{\text{uncond}} \leftarrow \theta(\mathcal{S}, t_k, c_{\text{neg}})$
  $v_{\text{cond}} \leftarrow \theta(\mathcal{S}, t_k, c_{\text{pos}})$
  $v \leftarrow v_{\text{uncond}} + w \cdot (v_{\text{cond}} - v_{\text{uncond}})$

  // Zero out predictions for condition frames
  $v[1] \leftarrow 0, \quad v[3] \leftarrow 0$

  // Scheduler step and restore conditions
  $\mathcal{S} \leftarrow \texttt{SchedulerStep}(v, t_k, \mathcal{S})$
  $\mathcal{S}[1] \leftarrow z_{\text{scr}}, \quad \mathcal{S}[3] \leftarrow z_s$
end for

// Decode each frame independently
for $n = 1$ **to** N **do**
  $\hat{x}_n \leftarrow D(\mathcal{S}[n])$
end for

return $\hat{x}_N$ *// final object-removed result*