Hint, Not Hole: Localization-Robust Video Overlay Removal

Technical Report

Dongyang Zhang, Wei Zhang, Renjie Zou, and Zhiwei Huang
Malanshan Audio & Video Laboratory
zhangdongyang@mlslabs.com.cn

Project page: https://zdyshine.github.io/HintErase



Figure 1: **One model for the overlays in a release.** HintErase removes karaoke lyrics with countdown dots over a crowded street, a multi-line title card together with the label logo, theme-song lyrics over fast animation, and credit lines in mixed fonts, from standard-definition sources upward. The hint (orange) is one box per text block or logo; the model sees the whole frame with the overlay still in place. All four clips play on the project page (#ktv-lyrics, #ktv-title, #anime-lyrics, #anime-credits).

Abstract

Video overlay removal recovers the scene beneath burned-in subtitles, karaoke lyrics, titles, credits, logos and watermarks. Generative video models now fill a known region convincingly, but in practice the region comes from a detector or a segmenter, and it is wrong in both directions: it misses strokes, halos, whole frames and whole objects, and it covers background between characters, around boxes and on genuine scene text. Erasers that treat the mask as a hole, regenerating inside and copying outside, inherit every one of these errors. We present HintErase, which treats localization as a hint, not a hole. The model sees the full, unmasked video together with a hint channel; during training the hint is corrupted the way real localizers fail, while supervision follows the true overlay. The result is a single model that removes overlay the hint missed, keeps background the hint wrongly covered, and is trained to keep genuine scene text even when the hint lands on it. For full-length releases, a streaming pipeline handles interlaced sources, shot cuts, missed and jittering hints, paste-back and audio, with memory independent of the length of the video. This report describes the approach, the pipeline that turns it into full-length deliveries, and qualitative results; video results, with input, hint and output side by side, are on the project page.

1 Introduction

Overlays are composited onto video after capture: hard subtitles, karaoke lyrics, title cards, end credits, station logos and watermarks. Removing them is a prerequisite for re-releasing film libraries, series and music videos in new languages and on new platforms. The task is demanding in ways that ob-

ject removal is not. Overlays are thin, outlined, glowing or semi-transparent; they appear, change color and vanish with the soundtrack; and one release carries thousands of overlaid frames that must all be erased without a visible seam.

Diffusion-based video models have made the *filling* half of the problem largely tractable [1–3]. Running such models on complete releases, we found that most remaining failures traced back to *localization* instead. Residue survived wherever the segmenter missed an outline or a glow, wherever the detector skipped a frame, and on small elements it never detected. Dilating the mask to catch these cases made the model repaint clean background between characters, which blurred texture and introduced flicker. The bottleneck had moved from how to fill to where to fill.

Existing methods resolve the question in one of two ways. *Mask-as-hole* erasers [1, 2, 4] regenerate the masked region and copy everything else, so any overlay pixel outside the mask survives, whatever the generator (Section 3). *Mask-free* subtitle removers [5, 6] need no mask, but cannot be told where to act and may erase genuine text in the scene. Both assume that the mask is either exact or absent.

We argue that localization should be read as a noisy observation of where overlays are, a *hint*, and not as the definition of where to generate, a *hole*. A remover built this way should be *complete*, removing overlay that the hint misses; *faithful*, keeping clean pixels that the hint covers; and *selective*, keeping genuine scene text and other overlay-like scene content even where the hint covers it.

Our contributions are:

- HINTERASE, a video overlay remover that conditions on the full video and a hint channel, and is trained with hints corrupted like real localizer output under supervision that follows the true overlay. One model accepts exact masks, loose boxes, partial masks or only a coarse region (Section 4).
- A streaming pipeline that turns the model into full-length deliveries: deinterlacing, shot segmentation, temporal hint repair, windowed inference with fixed memory, brightness-adaptive paste-back and frame-exact output with the original audio (Section 4.1).
- Qualitative results on karaoke lyrics, title cards with logos, theme-song lyrics and end credits in animation, imperfect hints and a multi-shot music video (Section 5); the video results play on the project page with input, hint and output side by side.

2 Related Work

Video inpainting and object removal. Propagation-based inpainters fill a mask by transferring visible content along time [4, 7, 8]. Diffusion models add a generative prior for content that is never revealed [1, 9], and diffusion transformers now lead object removal [2, 10–12]. These methods define the region to regenerate by the mask, which is the mask-as-hole design analyzed in Section 3.

Overlay and text removal. Image scene-text removal [13–15] and visible watermark removal [16–18] are mature for single images or a single overlay type. For video subtitles, CLEAR [5] and SEDiT [6] remove text without a mask, the latter in one sampling step with chunk-wise streaming for long videos. DeOverlay [3] starts a flow from the source latent, aggregates visible source observations over time inside each window, and introduces DVTE-Bench with eight text-overlay categories. Like DeOverlay, we keep the source visible. Our focus is different: how removal should behave when the mask is wrong, and how a full release should be processed.

Long videos. Training-free long-video methods co-denoise overlapping windows and average them [19, 20], shift a queue of frames at increasing noise levels [21], or condition each chunk on the previous one [22]. We process full releases in a streaming pipeline whose memory is independent of the length of the video (Section 4.1).

3 Hint, Not Hole

Let $\mathbf{x}^{\mathrm{src}}$ be a video with overlays and $\mathbf{x}^{\mathrm{gt}}$ its clean counterpart. The *true* overlay region $\mathcal{G}$ contains every pixel an overlay affects, including glow, shadow and compression halo. A localizer returns a *hint* $\mathcal{H}$: a segmentation mask, a box, a user stroke, or only a coarse region. Real hints err in two directions, *under-coverage* $\mathcal{G} \setminus \mathcal{H}$ (missed strokes, halos, whole frames and whole objects) and *over-coverage* $\mathcal{H} \setminus \mathcal{G}$ (dilation margins, box corners, gaps between characters, false positives). A remover $f(\mathbf{x}^{\mathrm{src}}, \mathcal{H})$ is *complete* if $f \approx \mathbf{x}^{\mathrm{gt}}$ on all of $\mathcal{G}$, *faithful* if $f \approx \mathbf{x}^{\mathrm{src}}$ outside $\mathcal{G}$, and *selective* if it stays faithful on scene content that resembles an overlay, such as scene text, even where $\mathcal{H}$ covers it. Selectivity is the hardest case of faithfulness, because that content looks like what is being removed.

Mask-as-hole erasers keep the input outside the mask by construction, through blanked conditioning, latent blending or paste-back. That alone sets a floor on what survives.

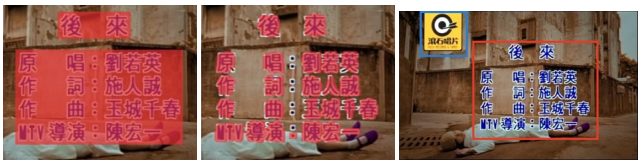
Proposition 1 (Residual floor)**.** *If $f(\mathbf{x}^{\mathrm{src}}, \mathcal{H}) = \mathbf{x}^{\mathrm{src}}$ outside $\mathcal{H}$, then on $\mathcal{G} \setminus \mathcal{H}$ the output equals the input, so the residual share*

$$\frac{\sum_{\mathcal{G}} |f(\mathbf{x}^{\mathrm{src}}, \mathcal{H}) - \mathbf{x}^{\mathrm{gt}}|}{\sum_{\mathcal{G}} |\mathbf{x}^{\mathrm{src}} - \mathbf{x}^{\mathrm{gt}}|} \;\geq\; \rho(\mathcal{G}, \mathcal{H}) = \frac{\sum_{\mathcal{G} \setminus \mathcal{H}} |\mathbf{x}^{\mathrm{src}} - \mathbf{x}^{\mathrm{gt}}|}{\sum_{\mathcal{G}} |\mathbf{x}^{\mathrm{src}} - \mathbf{x}^{\mathrm{gt}}|} \tag{1}$$

for every generator.

Dilating $\mathcal{H}$ lowers ρ only by enlarging $\mathcal{H} \setminus \mathcal{G}$, where the same erasers discard the input and must invent background they could have copied. Completeness and faithfulness therefore trade off by construction. Breaking the trade-off needs a model that may edit outside the hint and may copy inside it, i.e., one that reads the hint as evidence.

Figure 2 shows the trade-off on one title card. A region mask (a) covers whole lines, so a hole-filling eraser discards



(a) Region mask (b) Character mask (c) Box hint (HINTERASE)

Figure 2: **Three ways to localize the same title card.** (a) and (b) are too large and too small when read as a hole; (c) is coarser than both, yet sufficient when read as a hint. Red: text; blue: logo.

the clean background between and around the characters and has to repaint it. A character-level mask (b) follows the strokes, but outlines, glows and compression halos fall just outside it and survive, together with any stroke the segmenter missed. Read as a hint, one box per text block and per logo (c) is enough: the model sees the frame with the overlay in place, removes everything that belongs to the overlay, and copies the background inside the box.

4 HINTERASE

HINTERASE fine-tunes a pretrained video diffusion transformer trained with flow matching [23, 24] in a VAE latent space. Figure 3 shows a single window.

Hint as observation. The model always sees the full, unmasked video. Its input concatenates the noisy latent, the latent of the source video and an encoding of the hint,

$$\mathbf{z}_{\text{in}} = \big[\, \mathbf{z}_t \parallel \mathcal{E}(\mathbf{x}^{\text{src}}) \parallel \phi(\mathcal{H}) \,\big], \tag{2}$$

so the overlay pixels the hint misses and the clean pixels it covers both reach every layer. The hint encoder ϕ preserves how much of each latent cell the hint covers, so a tight character mask and a loose box over the same line stay distinguishable after downsampling. This tells the model what kind of localizer it is dealing with, and how far to trust it. With a weak hint, such as a coarse region instead of a mask, the same weights still remove the overlays inside it, so they do not depend on an exact mask.

Training with corrupted hints. Robustness is only learned for errors seen in training. The hint given to the model is therefore a corrupted copy of the true region, drawn from a model of the errors real front-ends make: tight masks, over-dilated masks, boxes, masks with missing strokes, frames or whole objects, and false positives, some of them placed on genuine scene text. Training clips are clean videos with overlays synthesized on the fly, so the clean target is known exactly everywhere.

Tri-region objective. Corrupting the input is not enough. If supervision followed the hint, the model would learn to regenerate exactly the hinted area and nothing else, which is the trade-off of Proposition 1 again. We weight the flow-matching loss by regions defined on the *truth*,

$$w = 1 + \lambda_G \tilde{G} + \lambda_H \max\big(H - \tilde{G},\, 0\big), \tag{3}$$

where H is $\mathcal{H}$ on the latent grid and $\tilde{G}$ is $\mathcal{G}$ on the latent grid, slightly dilated to include ringing. The overlay term covers all of $\mathcal{G}$ whether or not the hint does, which pays for completeness exactly where mask-as-hole erasers are blind. The hinted-background term covers $\mathcal{H} \setminus \mathcal{G}$, where the hint invites editing but the target equals the input, which pays for faithfulness. When a false-positive hint lands on scene text, the same term teaches selectivity.

4.1 From model to delivery

The model is trained and evaluated on clips of a few seconds; a delivery is a complete video of thousands of frames, and three gaps open between the two. *Short window, long video*: the video has to be split into windows and reassembled without seams, under a fixed memory budget. *One shot, frequent cuts*: a window that crosses a cut lets the shots on either side bleed into each other, while fades, flashes and dissolves look like cuts and are not. *Exact masks, imperfect localizers*: a frame the detector misses keeps its text, and a mask that jitters from frame to frame makes the output flicker. None of these shows on a short test clip; they surface together on a full-length video, and a failure in any one of them makes the delivery unusable. Table 1 lists the pipeline stage by stage, as a general method independent of any particular tuning; the specific values used to produce every result in this report are given separately below the table.

Values used for every result in this report: deinterlacing vote over the first 300 decoded frames; working size 448×1280 for subtitle bands and 576×1024 otherwise; shots of four frames or fewer merged into a neighbor; hint gaps of up to twelve frames bridged by a union within six frames on either side, hints extended by up to three frames at onset and offset; runs padded by eight frames of context and merged when at most 24 frames apart; 81-frame windows; paste-back threshold $\tau = \max\big(3,\ 12 \cdot \min(1, \bar{\ell}/96)\big)$, where $\bar{\ell}$ is the mean luma of the frame, applied within a 24-pixel neighborhood of the hint; $4\times$ upscaling for Ultra-HD delivery.

The pipeline needs no manual annotation: once the front-ends have produced hints, a release is processed end to end in a single run.

5 Results

Setup. Hints come from the production front-ends: a character-level subtitle segmenter and a logo and title-card detector. All masks and boxes in this report are computed by these models; Figure 4 additionally perturbs them, as described below. Except for the hint comparison in Figure 4, the hint is drawn as one box per text block or logo, in the figures and on the project page. All clips come from licensed productions and are shown with the permission of the rights holders. None of them were used in training; the training data was filtered to exclude them. We show HINTERASE

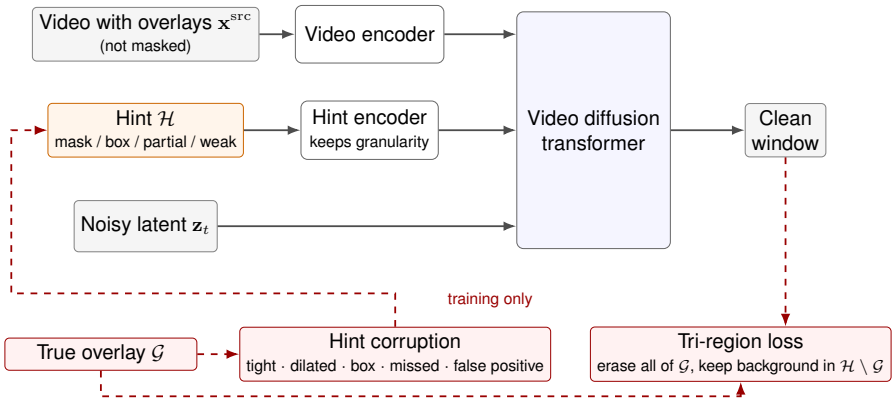


Figure 3: **Overview of HINTERASE.** The unmasked video and a separate hint channel condition a video diffusion transformer. In training, the hint is a corrupted copy of the true overlay region $\mathcal{G}$, while the loss is defined by $\mathcal{G}$ itself and by background the hint wrongly covers, so supervision never trusts the hint.

Stage	Problem	Solution
Input	Sources are interlaced, heavily compressed, variable-frame-rate, or stored with non-square pixels. Small text in low-resolution sources covers only a few latent cells.	Interlacing is detected by a majority vote over an initial window of decoded frames and removed with a field-parity-aware deinterlacer; the sample aspect ratio and frame timing are carried through to the output. Each run is cropped around the union of its hints and resized to a fixed working size chosen for the overlay type, so small text is magnified before encoding.
Shots	Windows that cross a cut mix two shots; fades, flashes and dissolves trigger false cuts; detectors leave fragments of a few frames.	A shot-boundary detector proposes cuts with a probability. Confident cuts are kept; the others are kept only if the frames on either side differ in both a gray thumbnail and a color histogram, with a looser test for low-probability cuts. Shots shorter than a minimum length are merged into a neighbor. Runs, windows and temporal hint extension never cross a remaining cut.
Hints	Segmenters miss frames, especially during fades; masks jitter; small marks such as logos and title cards are missed by a text segmenter.	The character-level subtitle mask is combined with boxes from a logo and title-card detector. Missed frames are filled in time: a gap between two hinted frames, up to a bounded length, is filled with the union of the hints within a fixed radius on either side, and a hint is extended by a bounded number of frames before an overlay first appears and after it last disappears. Hints need no spatial dilation, since the model reads them as evidence.
Planning	A full video does not fit in GPU memory; most frames carry no overlay at all.	A first decoding pass records frame times, thumbnails and per-frame hint statistics. Frames with a hint are grouped into runs, padded with context frames and merged when close enough together; only runs enter the model. Each run is covered by overlapping windows of fixed length. A second pass streams the video, and each frame is written as soon as no later window needs it; memory is bounded by the window length, independent of the length of the video.
Paste-back	Replacing whole windows softens untouched areas; residual text on dark frames differs from the background by only a few gray levels.	Only pixels the model changed are pasted back: a pixel is replaced if it differs from the input by more than an adaptive threshold that drops with the mean luma of the frame, so fades and night scenes are not under-replaced. The replaced region is restricted to a fixed neighborhood of the hint at the working resolution and feathered at its border.
Output	Dropped or duplicated frames; audio drifts out of sync.	Frames outside runs are written without entering the model, and the output has exactly as many frames as the source. The source audio is muxed back, compensated for the offset between the video stream and the container start, and trimmed to the video duration; truncating to the shorter stream instead can drop the last frame.
Ultra-HD	Old footage is low-resolution, while re-releases often require Ultra-HD; generative upscalers built on large diffusion models [25] can shift colors, alter faces and people, and invent details.	Optional: after erasure, the clean video is upscaled by a fixed factor with an in-house super-resolution model designed to preserve the original colors, people and details.

Table 1: **From model to delivery.** Each stage of the long-video pipeline, the problem it addresses on full-length releases, and how it is solved in general. Cost scales with the duration of the overlays, not with the length of the video.

only. The question this report asks is how removal should behave when the hint is wrong, and we answer it by running the same weights with different hints (Figure 4). Still frames cannot show flicker or seams; the clips in Figure 1 play on the project page, where input, hint and output run side by side on a shared timeline.



Figure 4: **The same weights under five hints.** Top: the hint (orange) on the input. Bottom: the output of HINTERASE, which reads the hint as evidence. A mask-as-hole eraser cannot change pixels outside the hint (Proposition 1), so it would leave the outlines just outside the character mask (a), the halo outside the eroded mask (b), the lower strokes of the last two characters (c), and all of the lyrics inside a detection box drawn as its outline (e); with a filled box (d) it would have to regenerate all the clean background inside the box. HINTERASE removes the lyrics under every hint.

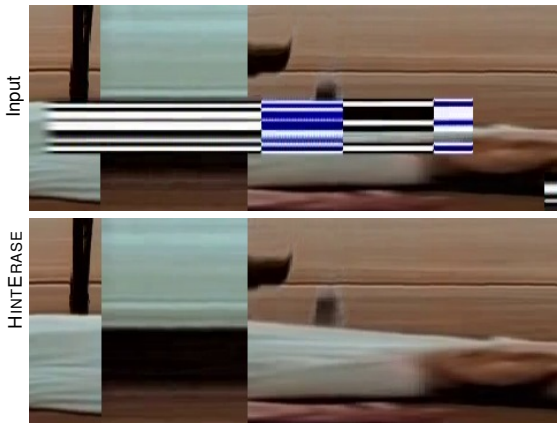


Figure 5: **Time slice through the lyrics band.** Each image is one pixel column through the lyrics of a karaoke music video, with 18 seconds of frames running left to right. In the input, lyric lines appear as striped blocks that change color as the wipe passes; in the output, the streaks of the background run through them without vertical steps.

Overlay types. Figure 1 shows one example for each of four overlay types. Karaoke lyrics are among the most demanding subtitle cases in our footage, because the fill changes color mid-line under an outline that does not, and countdown dots appear before each line (`#ktv-lyrics`). Title cards stack several lines of credits next to a label logo, and fade in and out with the shot (`#ktv-title`); the logo is hinted by its own box. Theme-song lyrics in animation sit on top of fast, saturated motion (`#anime-lyrics`), and credit lines at the end of an episode mix fonts and colors over busy scenes (`#anime-credits`). Low-resolution animation has flat colors and thin outlined text, which make any residue easy to see.

Imperfect hints. Figure 4 runs the same weights on one clip under five hints that a real localizer could produce: a character mask, the same mask eroded so that it misses the outlines, the same mask missing the lower 20% of the strokes of two characters where they cross a white garment, a filled box per line, and the outline of a detection box. Read as a hole, each of them fails in a predictable way: by Proposition 1, nothing outside the hint can change, so the dark outlines just outside even the character mask, the halos outside the eroded mask and the uncovered strokes all remain, and a detection box drawn as its outline would regenerate the outline and nothing else; a filled box covers the lyrics but would discard the clean background inside it. Read as a hint, the same pixels are visible to the model, and HINTERASE removes the lyrics under all five hints, so the localizer can be as coarse as a box per line, as in Figure 2(c).

Full-length videos. Figure 5 looks at a karaoke music video through a time slice: one pixel column through the lyrics band, with frames running left to right. A static background appears as horizontal streaks, a shot cut as a vertical edge, and a seam or flicker between windows as a vertical step where no cut is. The clip is cut from a complete music video processed in one run with the pipeline of Section 4.1, and the slice shows the lyrics band continuing the background across shot after shot without steps.

Ultra-HD delivery. Old karaoke footage is often re-released in Ultra-HD, and upscaling is where a re-release most easily loses fidelity: generative video upscalers built on large diffusion models, such as FlashVSR+ [25], can shift colors, alter faces and people, and invent texture. The pipeline therefore erases the overlays at the source resolution and then upscales the clean video $4\times$ with an in-house super-resolution model designed for fidelity rather than generation (Table 1). Upscaling also magnifies whatever the eraser leaves behind, so the erased regions have to stay consistent with their surroundings at the delivered resolution.

6 Conclusion

Conclusion. Generative models have made filling a known region the easy part of overlay removal; deciding what the region is remains hard, and practical localizers err in both directions. HINTERASE treats the localizer's output as a hint: it observes the full video, learns from hints corrupted like real ones, and is supervised by the truth. For full releases, a streaming pipeline handles deinterlacing, shot cuts, imperfect hints, paste-back and audio, so a complete video is pro-

cessed in one run. The model does not need a better localizer, only a plausible one.

Broader impact. Overlay removal can be misused to strip watermarks or attribution from content one has no right to modify. HINTERASE was developed to re-release licensed content; deployments should keep provenance metadata and restrict watermark removal to rights holders.

References

[1] Xiaowen Li, Haolan Xue, Peiran Ren, and Liefeng Bo. DiffuEraser: A Diffusion Model for Video Inpainting. *arXiv preprint arXiv:2501.10018*, 2025.

[2] Bojia Zi, Weixuan Peng, Xianbiao Qi, Jianan Wang, Shihao Zhao, Rong Xiao, and Kam-Fai Wong. MiniMax-Remover: Taming Bad Noise Helps Video Object Removal. In *NeurIPS*, 2025.

[3] Deshui Miao, Yuhang Ma, Shaoyong Jia, Wenzhao Gao, Yue Zhang, Shaohui Jiao, and Xin Li. DeOverlay: A Benchmark and Source-Anchored Flow for Video Overlay Removal. *arXiv preprint arXiv:2610.26081*, 2026.

[4] Shangchen Zhou, Chongyi Li, Kelvin C. K. Chan, and Chen Change Loy. ProPainter: Improving Propagation and Transformer for Video Inpainting. In *ICCV*, 2023.

[5] Qingdong He, Chaoyi Wang, Peng Tang, Yifan Yang, and Xiaobin Hu. CLEAR: Context-Aware Learning with End-to-End Mask-Free Inference for Adaptive Video Subtitle Removal. In *ICML*, 2026.

[6] Zheng Hui and Yunlong Bai. SEDiT: Mask-Free Video Subtitle Erasure via One-step Diffusion Transformer. *arXiv preprint arXiv:2605.14894*, 2026.

[7] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning Joint Spatial-Temporal Transformations for Video Inpainting. In *ECCV*, 2020.

[8] Zhen Li, Cheng-Ze Lu, Jianhua Qin, Chun-Le Guo, and Ming-Ming Cheng. Towards An End-to-End Framework for Flow-Guided Video Inpainting. In *CVPR*, 2022.

[9] Zhixing Zhang, Bichen Wu, Xiaoyan Wang, Yaqiao Luo, Luxin Zhang, Yinan Zhao, Peter Vajda, Dimitris Metaxas, and Licheng Yu. AVID: Any-Length Video Inpainting with Diffusion Model. In *CVPR*, 2024.

[10] Yuxuan Bian, Zhaoyang Zhang, Xuan Ju, Mingdeng Cao, Liangbin Xie, Ying Shan, and Qiang Xu. VideoPainter: Any-length Video Inpainting and Editing with Plug-and-Play Context Control. *arXiv preprint arXiv:2503.05639*, 2025.

[11] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-One Video Creation and Editing. *arXiv preprint arXiv:2503.07598*, 2025.

[12] Jie Liu and Zheng Hui. EraserDiT: Fast Video Inpainting with Diffusion Transformer Model. *arXiv preprint arXiv:2506.12853*, 2025.

[13] Chongyu Liu, Yuliang Liu, Lianwen Jin, Shuaitao Zhang, Canjie Luo, and Yongpan Wang. EraseNet: End-to-End Text Removal in the Wild. *IEEE Transactions on Image Processing*, 29:8760–8775, 2020.

[14] Chongyu Liu, Lianwen Jin, Yuliang Liu, Canjie Luo, Bangdong Chen, Fengjun Guo, and Kai Ding. Don't Forget Me: Accurate Background Recovery for Text Removal via Modeling Local-Global Context. In *ECCV*, 2022.

[15] Dezhi Peng, Chongyu Liu, Yuliang Liu, and Lianwen Jin. ViTEraser: Harnessing the Power of Vision Transformers for Scene Text Removal with SegMIM Pretraining. In *AAAI*, 2024.

[16] Yang Liu, Zhen Zhu, and Xiang Bai. WDNet: Watermark-Decomposition Network for Visible Watermark Removal. In *WACV*, 2021.

[17] Xiaodong Cun and Chi-Man Pun. Split Then Refine: Stacked Attention-Guided ResUNets for Blind Single Image Visible Watermark Removal. In *AAAI*, 2021.

[18] Junye Chen, Chaowei Fang, Jichang Li, Yicheng Leng, and Guanbin Li. Decouple and Couple: Exploiting Prior Knowledge for Visible Video Watermark Removal. *IEEE Transactions on Image Processing*, 34:1192–1203, 2025.

[19] Fu-Yun Wang, Wenshuo Chen, Guanglu Song, Han-Jia Ye, Yu Liu, and Hongsheng Li. Gen-L-Video: Multi-Text to Long Video Generation via Temporal Co-Denoising. *arXiv preprint arXiv:2305.18264*, 2023.

[20] Shuangquan Lyu, Steven Mao, and Yue Ma. Unified Long Video Inpainting and Outpainting via Overlapping High-Order Co-Denoising. *arXiv preprint arXiv:2511.03272*, 2025.

[21] Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. FIFO-Diffusion: Generating Infinite Videos from Text without Training. In *NeurIPS*, 2024.

[22] Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. StreamingT2V: Consistent, Dynamic, and Extendable Long Video Generation from Text. In *CVPR*, 2025.

[23] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *ICLR*, 2023.

[24] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *ICLR*, 2023.

[25] Ruchang Yao, Shi Guo, Junhao Zhuang, Mingde Yao, Xin Cai, Jiahao Zhan, Bin Fu, Xinyuan Chen, Yu Qiao, and Tianfan Xue. FlashVSR+: Efficient Video Super-Resolution via Hybrid Attention and Reward-Yield Distribution Matching. *arXiv preprint arXiv:2610.17552*, 2026.