DECACHE: DECOUPLED AND DAMPED MEMORY PURIFICATION FOR ROBUST STREAMING VIDEO ENHANCEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming Video Super-Resolution (VSR) relies on causal Key-Value (KV) memory for constant-memory, low-latency inference, but suffers critically from causal cache contamination, where corrupted frames pollute the historical context, causing irreversible error accumulation and hallucination drift. To address this, we propose DeCache, which introduces a Read-Write Decoupled Routing mechanism to separate the dual roles of each frame: raw degraded inputs anchor current-frame generation for physical fidelity, while model-restored "pseudo-clean" representations update the temporal cache to protect future memory. Through a causal Gauss-Seidel/Jacobi analysis, we prove both schedules converge exactly to the same causal fixed point, and show that one round decoupling is empirically Pareto-optimal, whereas converging to the fixed point triggers self-reinforcing hallucination loops and over-sharpening. We further design an Asymmetric Damped KV scheme, using a convex combination for Values to calibrate the perception-distortion trade-off. Finally, a 1-chunk lookahead asynchronous streaming architecture achieves mathematical equivalence to offline one-step decoupling in single-pass online inference. Experiments show DeCache sets new state-of-the-art results in perceptual quality while maintaining competitive full-reference fidelity and temporal consistency.

1 INTRODUCTION

Video super-resolution (VSR) aims to reconstruct high-resolution videos from low-resolution observations by exploiting information across frames. Diffusion-based VSR methods exploit generative priors to reconstruct plausible details under complex degradations (Zhou et al., 2024; Yang et al., 2024; Xie et al., 2025; Wang et al., 2026). Meanwhile, advances in efficient video diffusion have made streaming restoration increasingly practical (Zhuang et al., 2026; Zhang et al., 2025; Bai et al., 2026). In a streaming setting, a model processes incoming video chunks with bounded temporal context and carries information forward through persistent states, such as a key–value (KV) cache. This memory supports temporal information reuse without processing the entire video at once. Its contents, however, also determine how errors in earlier observations influence later reconstructions.

A central challenge is that memory updates are typically coupled to current-chunk reconstruction. The degraded observation used to reconstruct a chunk also contributes to the states stored for future chunks. Consequently, the cache may preserve degradation alongside useful scene information. This problem becomes particularly visible after a localized corruption: although the incoming observations have recovered, the model can still attend to states formed from damaged content. We refer to this failure mode as causal cache contamination. It motivates a distinction between two roles of the input: providing evidence for the current reconstruction and providing reliable context for future reconstructions.

Using restored outputs to construct memory offers a natural remedy, but introduces a competing risk. A model-generated estimate may suppress degradation while also adding inaccurate textures or altering image structures. Feeding this estimate back as the input to reconstruction can weaken the connection to the original observation and reinforce the model's own errors. Restoration therefore faces a tension: the raw observation preserves measurement evidence but may contaminate memory,

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.

whereas a restored estimate can improve memory quality but carries generative uncertainty. Addressing this tension requires control over both where feedback enters the model and how strongly it influences subsequent predictions.

We propose DECACHE, a training-free framework that decouples current-chunk rendering from temporal memory writing. For each chunk, a rendering pass retains the original low-resolution observation, while a separate memory pass uses a downsampled restored estimate as pseudo-clean conditioning. Because pseudo-clean estimates are imperfect, DeCache further damps this feedback through a key/value routing rule that separately controls memory addressing and retrieved content, trading artifact suppression against fidelity.

A further question is how far this feedback should be propagated. We formulate restoration-driven memory updates as a causal trajectory operator and distinguish recursive feedback from refinement using a fixed draft trajectory. Repeated application seeks a self-consistent result, but self-consistency does not guarantee reconstruction fidelity: model-generated details can become evidence for subsequent predictions. Our ablations show that further refinement can reduce full-reference fidelity even as some perceptual scores increase. These observations motivate a single refinement round seeded by the unmodified backbone, which limits the repeated reuse of generated content.

Although this refinement can be defined using a precomputed video, it does not require full-video storage at deployment. DeCache uses a leading draft stream to generate pseudo-clean conditions and a following rendering stream to consume them. With the required chunk offset and matched inference conditions, this schedule preserves the dependencies of offline one-round refinement while keeping memory and buffering independent of video length. We instantiate DeCache on FlashVSR (Appendix F). DeCache requires approximately three backbone forward passes per streaming step and runs at $0.45\times$ baseline throughput with unchanged lookahead latency. Overall, our contributions can be summarized as follows: **(i)** We formulate causal cache contamination as a memory-update problem in streaming VSR and introduce a training-free separation of observation-conditioned rendering and restoration-conditioned memory writing; **(ii)** We introduce damped value feedback and analyze the distinction between recursive self-conditioning and draft-seeded refinement, motivating controlled feedback rather than unrestricted iteration; **(iii)** We provide a 1-chunk lookahead asynchronous dual-stream architecture that reproduces offline one-round refinement in single-pass online inference, and show state-of-the-art perceptual quality at competitive full-reference fidelity and temporal consistency.

2 RELATED WORK

Recurrent Video Restoration and Propagation Stability. Video super-resolution exploits temporal information through alignment, recurrent propagation, and attention, as demonstrated by EDVR, BasicVSR/BasicVSR++, and VRT/RVRT (Wang et al., 2019; Chan et al., 2021; 2022a; Liang et al., 2024; 2022). FRVSR explicitly feeds back the preceding HR estimate, after motion compensation and a space-to-depth transformation, together with the current LR frame (Sajjadi et al., 2018; Fuoli et al., 2023). Stability under recurrent feedback has also been studied: TecoGAN introduces a Ping-Pong training loss to reduce temporal artifact accumulation (Chu et al., 2020a), while Chiche et al. study instability on long, low-motion sequences and introduce MRVSR, which confines Lipschitz constraints to its recurrent subnetwork to preserve the expressiveness of input and output networks (Chiche et al., 2022). For real-world degradations, RealBasicVSR introduces input pre-cleaning to mitigate degradation amplification through propagation (Chan et al., 2022b), and FastRealVSR addresses artifacts carried by hidden states using a filtered state pool and input-guided selective attention (Xie et al., 2023). These studies address reconstruction feedback and propagation stability through learned architectures or training objectives. DeCache instead investigates how to construct KV memory in a frozen streaming diffusion backbone, separating the LR conditions used for current-chunk reconstruction and memory writing.

Streaming Video Diffusion and Cache Construction. Diffusion-based restoration methods, including STAR, DOVE, and SeedVR2, exploit generative video priors (Xie et al., 2025; Chen et al., 2026; Wang et al., 2026). CausVid and Self Forcing study efficient autoregressive video diffusion and training with model-generated histories, respectively (Yin et al., 2025; Huang et al., 2026). Both methods re-evaluate denoised outputs at zero noise to construct historical KV states. In

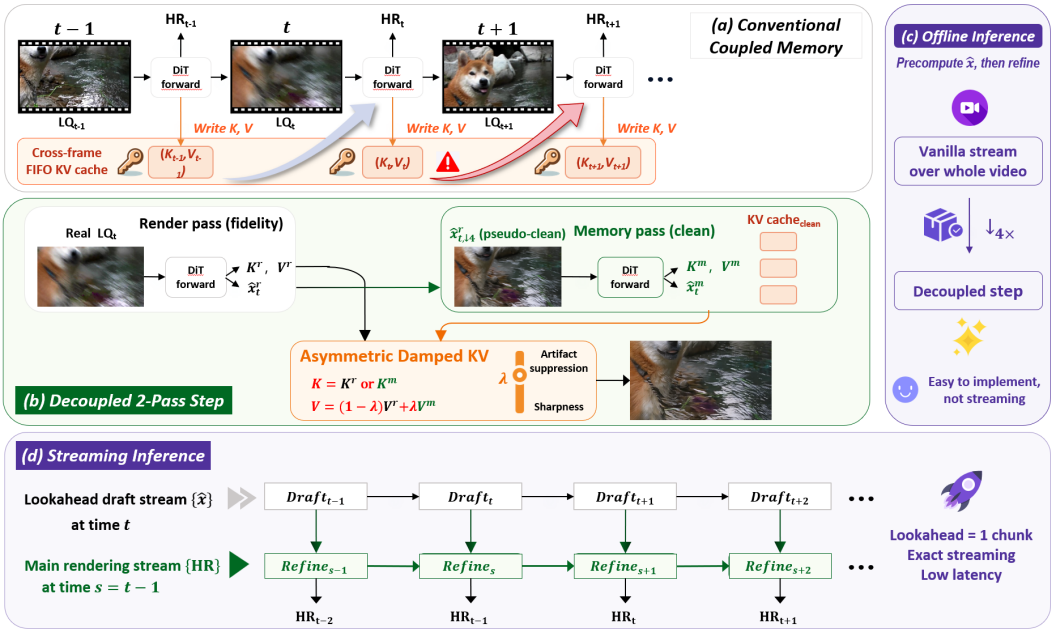

Figure 1: **Overview of DeCache.** (a) In a conventional streaming backbone, one forward pass both renders a chunk and writes its KV states, so degraded inputs contaminate the cache. (b) The render pass uses the real LQ input, while the memory pass uses a pseudo-clean input $\tilde{x}_t$; keys are routed and values mixed with weight λ to trade artifact suppression against sharpness. (c) Offline: precompute $\tilde{x}$ with the vanilla backbone, then run one decoupled pass. (d) Streaming: a draft stream runs one chunk ahead of the rendering stream, reproducing the offline result exactly with bounded latency and memory.

FlashVSR, however, cache construction also depends on the LR condition, motivating DeCache to modify this condition for memory writing while retaining the original observation for rendering. FlashVSR develops a one-step streaming restoration model that constructs temporal KV states during LR-conditioned inference (Zhuang et al., 2026). Closely related systems also include InfVSR for streaming VSR and InstantViR for causal video restoration (Zhang et al., 2025; Bai et al., 2026), whose conditioning pathways and cache designs are discussed in Appendix F. Cache construction has also been treated separately from output generation: Ca^2-VDM performs a cache-writing evaluation on denoised latents to obtain KV states shared across subsequent denoising steps (Gao et al., 2024). Other methods revise historical context for different purposes. LongLive rebuilds KV states using generated video and a new prompt to support prompt transitions (Yang et al., 2025), while Steady-Forcing periodically flushes the cache to mitigate accumulated errors in long-horizon nature video generation (Minar et al., 2026). DeCache instead targets memory construction under degraded visual observations. It obtains candidate memory states by changing the LR conditioning of an additional forward pass, while keeping its noise and historical KV cache identical to those of the real-LR rendering pass. This conditions the two computations differently without modifying backbone parameters.

Prediction Feedback and Reconstruction Fidelity. Self-conditioning reuses a previous prediction as additional input to a diffusion model (Chen et al., 2022). Such feedback is informative but may contain model errors. In restoration, the perception–distortion trade-off further distinguishes perceptual realism from reference fidelity (Blau & Michaeli, 2018). DeCache constructs pseudo-clean conditions from a separately computed baseline trajectory and uses one outer refinement round, rather than recursively seeding memory with the refined outputs. The rendering path retains the observed LR input; key-source selection and convex value mixing control which candidate states are committed to memory. We analyze the relationship between draft-seeded refinement and recursive feedback, while treating reconstruction quality as an empirical question rather than a consequence of fixed-point convergence.

3 METHOD

3.1 PROBLEM FORMULATION

Streaming restoration. Given a degraded low-resolution video of L frames, we seek its clean high-resolution counterpart. A block-causal streaming backbone partitions the sequence into T chunks and emits them one at a time; we write $x = (x_1, \dots, x_T)$ and $y = (y_1, \dots, y_T)$ for the chunk-wise LR input and HR output, so that chunk t is produced by a single denoising step

$$y_t \;=\; M\big(x_t\,;\,\mathcal{C}_{<t}\big), \qquad \mathcal{C}_{<t} \;=\; \Phi\big(\{h_s\}_{s\in W_t}\big), \tag{1}$$

where x_t is the real-LR conditioning of chunk t, $h_s = (K_s, V_s)$ are the key/value states written by chunk s, $W_t = \{s : \max(1, t-k) \le s < t\}$ is the strictly-past window of the length-k FIFO cache, and Φ is the block-sparse causal attention that assembles them into the memory $\mathcal{C}_{<t}$.

Here M denotes one chunk of the streaming backbone as a black box: it encodes the LR conditioning, runs a single denoising step of the DiT F in latent space, and decodes the result. Unrolling it exposes the memory write we will later modify. With $\boldsymbol{z}_t$ the noised latent of chunk t, $\boldsymbol{c}_t = P(x_t)$ its condition tokens produced by the condition projection P, and $\boldsymbol{v}_t$ the predicted velocity,

$$\big(\boldsymbol{v}_t,\, h_t\big) \;=\; F\big(\boldsymbol{z}_t;\, \boldsymbol{c}_t,\, \mathcal{C}_{<t}\big), \qquad y_t \;=\; \mathrm{VAE.dec}\big(\boldsymbol{z}_t - \boldsymbol{v}_t\big), \tag{2}$$

so a single forward pass both renders chunk t and writes its memory: h_t is produced on-the-fly, inside the block stack, from the very conditioning $\boldsymbol{c}_t$ that renders the chunk. This coupling is what DECACHE breaks. The cache is a fixed-length FIFO, giving bounded-latency, bounded-memory streaming.

Quality-blind memory propagates degradation. The write in Eq. 2 is unconditional on quality: chunk s commits h_s to the cache whatever its rendering turned out to be, so a chunk whose conditioning yields a poor y_s injects suboptimal key/value states that every chunk reading s then attends to. The effect is general, but it is sharpest—and cleanest to isolate—under a localized corrupted sub-interval $\mathcal{S} = \{s : x_s \text{ corrupted}\}$ (e.g. tape tearing, bitstream errors), which contaminates the memory of every chunk whose window still overlaps it:

$$\mathcal{C}_{<t} \text{ carries the corruption} \quad\Longleftrightarrow\quad W_t \cap \mathcal{S} \neq \varnothing, \tag{3}$$

i.e. for all t with $\min \mathcal{S} < t \le \max \mathcal{S} + k$. The damage thus outlives the corrupted input by up to a full cache length. It is most visible on the recovery chunk $t^\star = \max \mathcal{S} + 1$: its conditioning $x_{t^\star}$ is already clean, yet $\mathcal{C}_{<t^\star}$ still holds $\{h_s\}_{s\in\mathcal{S}}$, so the output inherits artifacts the input no longer has. This makes $t^\star$ a controlled probe of the memory path alone, the rendering conditioning there is identical across all methods we compare, so any difference in $y_{t^\star}$ is attributable to what was written into the cache. Our objective is a memory-writing rule that keeps $\mathcal{C}$ clean while leaving the rendering conditioning x_t untouched, so that recovery and downstream chunks improve without sacrificing faithful reconstruction on clean regions.

From corruption detection to an unconditional rule. A natural first instinct is to detect corrupted chunks and special-case their memory writes: identify $s \in \mathcal{S}$, then substitute a restored pseudo-clean observation for h_s. This conditional design, however, only covers the symptom it was built to catch. Memory poisoning is not specific to explicit corruption: any chunk whose conditioning yields a suboptimal rendering—whether from tape damage, blur, difficult texture, or accumulated drift—still writes a suboptimal h_s that propagates downstream. Explicit corruption is simply the sharpest, most easily isolated instance of a broader failure mode, not its only cause. We therefore drop the detection step entirely and adopt an unconditional rule: every chunk's memory is written from the model's own restored output rather than its raw conditioning, regardless of whether that chunk is corrupted. This removes the need for a corruption detector (itself a fragile sub-problem prone to missed and false detections) and additionally guards against silent degradation that a hard corruption label would miss.

Algorithm 1 DeCache

Require: LR $\boldsymbol{x}$; DiT F; condition proj P; steps T; cache len k; pseudo-clean cond $\tilde{\boldsymbol{x}} = \downarrow_4 (\boldsymbol{y}^{(0)})$ (vanilla backbone output); mix λ; key$\in \{\text{real}, \text{mem}\}$

$\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$; $\mathcal{C} \leftarrow \varnothing$
for $t = 0$ **to** $T - 1$ **do**
  $\boldsymbol{c}_t \leftarrow P(\boldsymbol{x}_t)$
  $\tilde{\boldsymbol{c}}_t \leftarrow P(\tilde{\boldsymbol{x}}_t)$ ▷ $\tilde{\boldsymbol{x}}$ condition (for MEMORY pass)
  $(\boldsymbol{v}_t, K_t^r, V_t^r) \leftarrow F(\boldsymbol{z}_t; \boldsymbol{c}_t, \mathcal{C})$ ▷ OUTPUT pass: real LR
  $(\cdot, K_t^m, V_t^m) \leftarrow F(\boldsymbol{z}_t; \tilde{\boldsymbol{c}}_t, \mathcal{C})$ ▷ MEMORY pass: $\tilde{\boldsymbol{x}}$
  $K_t \leftarrow \begin{cases} K_t^r & \text{key} = \text{real} \\ K_t^m & \text{key} = \text{mem} \end{cases}$
  $V_t \leftarrow (1-\lambda) V_t^r + \lambda V_t^m$ ▷ Mixed-V
  $\mathcal{C} \leftarrow \text{FIFO}_k(\text{append}(\mathcal{C}, (K_t, V_t)))$
  $\hat{\boldsymbol{z}}_t \leftarrow \boldsymbol{z}_t - \boldsymbol{v}_t$
end for
return $\text{VAE.dec}(\{\hat{\boldsymbol{z}}_t\})$

3.2 Output/Memory Decoupling

Decoupled rendering and memory paths. We split the single LR input into two roles: the rendering path keeps the real LR x, while only the memory path (the KV-cache write) is seeded by a pseudo-clean feedback $\tilde{x}$, obtained by re-degrading a separately computed baseline restoration, $\tilde{x}_t = \downarrow_4 (y_t^{(0)})$, where $y^{(0)}$ denotes the trajectory of the undecoupled, vanilla backbone (Sec. 3.4). Because $y^{(0)}$ is computed independently of the decoupled output, there is no circularity between rendering and memory: the memory each chunk reads was written by a stream that never saw the decoupled output at all. Formally, Eq. 2 is re-sourced as two forwards over the same conditioning port:

$$\big(\boldsymbol{v}_t,\, K_t^r,\, V_t^r\big) = F\big(\boldsymbol{z}_t;\, \boldsymbol{c}_t,\, \mathcal{C}_{<t}\big), \big(\cdot,\, K_t^m,\, V_t^m\big) = F\big(\boldsymbol{z}_t;\, \tilde{\boldsymbol{c}}_t,\, \mathcal{C}_{<t}\big), \boldsymbol{c}_t = P(x_t), \tilde{\boldsymbol{c}}_t = P(\tilde{x}_t). \quad (4)$$

The two passes read the identical memory and differ only in conditioning: the render pass supplies the emitted velocity $\boldsymbol{v}_t$, decoded into y_t; the memory pass supplies a second, independently-conditioned key/value pair. Both pairs contribute to $h_t = (K_t, V_t)$; Sec. 3.3 defines the combination rule.

Two forwards are necessary rather than incidental: each block's K, V are computed on-the-fly from that block's own input activations, which descend jointly from $\boldsymbol{z}_t$ and its conditioning. A clean h_t therefore requires $\tilde{x}_t$ to propagate through the entire stack before its K, V are read—a requirement no post-hoc substitution at the cache boundary can satisfy, since the backbone exposes a single conditioning port per traversal. The only way to obtain both a clean h_t and a faithful $\boldsymbol{v}_t$ is to traverse the stack twice. The cost is one extra forward per chunk; the cache structure itself is unchanged, so the memory footprint matches the baseline. Algorithm 1 gives the full procedure.

3.3 Asymmetric Damped KV: Trading Artifact Suppression for Sharpness

Both pairs contribute to $h_t = (K_t, V_t)$, combined as

$$K_t \;=\; \begin{cases} K_t^r & \text{key} = \text{real} \\ K_t^m & \text{key} = \text{mem} \end{cases}, \qquad V_t \;=\; (1-\lambda)\, V_t^r + \lambda\, V_t^m, \qquad h_t = (K_t, V_t), \qquad (5)$$

Keys set memory addressing; values set the retrieved content once that address is fixed. Our Asymmetric Damped KV scheme routes K_t discretely between the two passes and damps V_t continuously with a single scalar $\lambda \in [0, 1]$, since $\tilde{x}_t$ is itself an imperfect estimate and need not be trusted outright. $\lambda \to 0$ keeps V_t close to the real-LR rendering; $\lambda \to 1$ leans on the pseudo-clean memory pass. We fix key and λ per dataset from a sweep (Fig. 3, Appendix B.1). Convex value mixing is motivated by MSE–W_2 perception–distortion theory (Freirich et al., 2021).

3.4 THE FEEDBACK IS A RELAXATION; ITS FIXED POINT IS THE WRONG TARGET

Collecting all chunks into $Y = (y_1, \ldots, y_T)$ turns the per-chunk decoupling of Eq. 4 into a whole-trajectory operator,

$$\big(\mathcal{M}(Y)\big)_t \;=\; M\Big(x_t;\, \Phi\big(\{\downarrow_4 (y_s)\}_{s\in W_t}\big)\Big). \tag{6}$$

This operator admits two schedules:

- **Online, single pass**: one causal sweep in which chunk t's memory draws on chunks $s < t$ already decoded earlier in the same sweep. This is a Gauss–Seidel iteration of $\mathcal{M}$.
- **Offline, N rounds**: round k freezes the entire previous trajectory as the memory seed, $Y^{(n)} = \mathcal{M}(Y^{(n-1)})$, with $Y^{(0)}$ the trajectory of the undecoupled, vanilla backbone (Eq. 2 run without decoupling). This is a Jacobi, or Picard, iteration.

Both schedules solve the same self-consistency condition. Because each chunk reads only strictly-past memory, this condition has a unique solution $Y^\star = \mathcal{M}(Y^\star)$, which the offline schedule reaches exactly after finitely many rounds and the online schedule computes in a single sweep, with no assumption on $\mathcal{M}$ (Proposition 1 and Proposition 2, Appendix C). The two schedules are therefore the same trajectory computed in two ways, not two distinct methods; Fig. 3 confirms this empirically, with successive offline rounds approaching the online result. Crucially, this shared fixed point is itself a fidelity-degrading target: full-reference fidelity decreases monotonically with each round, and iterating toward convergence drives the output toward an over-sharpened, hallucinated solution (Fig. 3 and Fig. 6, Appendix C.3). We therefore never iterate to convergence. Our configuration is the first Jacobi iterate, it_1: $Y^{(1)} = \mathcal{M}(Y^{(0)})$, a single decoupled pass seeded by the vanilla trajectory. It retains the clean-memory recovery gain and attains the best aesthetic-oriented scores at a fraction of the fidelity cost of further iteration. The optimal operating point is therefore the earliest informative iterate, not the limit.

3.5 DEPLOYMENT: AN EXACT STREAMING REALIZATION OF it_1

Although it_1 is defined offline, its seed $Y^{(0)}$ precomputed by running the vanilla backbone once over the entire video, it can be realized exactly as a single causal stream. The key is that $Y^{(0)}$ depends only on the vanilla stream's own real-LR cache $\mathcal{C}^{\mathrm{base}}_{<t}$ and is independent of the decoupled output $Y^{(1)}$, so there is no output↔memory circular dependency between the two streams. Running the vanilla stream one chunk ahead of the decoupled stream therefore supplies every memory frame in time: by the time chunk t's decoupled pass needs $\downarrow_4 (y_t^{(0)})$, the vanilla stream has already produced it. The decoupled rendering thus always reads a clean past memory, with no approximation (Algorithm 2). This yields a deployable single pass with constant latency and bounded memory, whose output matches the offline it_1 frame-for-frame. DECACHE assumes a backbone with a managed streaming KV cache and a persistent LQ conditioning pathway; we characterise this class in Appendix F.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and Base Model. We evaluate DeCache on five benchmarks: TAPE (Agnolucci et al., 2024), BSCV (Liu et al., 2023), BSD (Zhong et al., 2023), AbsoluteDegradation (Jastrzebski et al., 2026), and SRWOV/real (Mao et al., 2025). We select benchmarks with severe and heterogeneous degradations to assess robustness (Appendix B.2), where unreliable observations can compromise both current-frame reconstruction and the temporal memory used for subsequent frames. Representative examples are shown in Fig. 4. All experiments are conducted at a $4\times$ upscaling factor. We instantiate DeCache on the pretrained FlashVSR (Zhuang et al., 2026) backbone without additional training or parameter updates. Detailed dataset configurations, preprocessing procedures, and inference settings are provided in Appendix B.2.



Figure 2: Qualitative comparison for video super-resolution on real-world videos.

Baselines. We compare DeCache with STAR (Xie et al., 2025), DOVE (Chen et al., 2026), SeedVR2 (Wang et al., 2026), FlashVSR, and InstantViR (Bai et al., 2026), covering representative diffusion-based video super-resolution and restoration methods. The unmodified FlashVSR backbone serves as the primary baseline for isolating the effect of the proposed memory-update mechanism.

Evaluation Metrics. We evaluate three complementary aspects: (1) reconstruction fidelity: PSNR and SSIM; (2) perceptual quality: LPIPS (Zhang et al., 2018), NIQE (Mittal et al., 2012), MUSIQ (Ke et al., 2021), CLIP-IQA (Wang et al., 2023), FasterVQA (Wu, 2022), and DOVER (Wu et al., 2023); and (3) temporal consistency: E_{warp} (Lai et al., 2018) and tOF (Chu et al., 2020b). Reference-based metrics are reported only on benchmarks with GT.

4.2 Comparison with State-of-the-Art Methods

Quantitative Results. As shown in Tab. 1, DeCache maintains fidelity closely competitive with existing approaches on paired-reference benchmarks, within a modest margin of regression-oriented methods such as DOVE and STAR, despite the latter being explicitly optimized for pixel-level reconstruction. This confirms that our method introduces substantially stronger generative capability without a disproportionate cost to reconstruction accuracy. For perceptual quality, DeCache achieves top-1 or top-2 performance across the majority of datasets and no-reference metrics, with a particularly clear margin on the real-world, no-reference benchmarks, demonstrating strong generalization to authentic degradations beyond the synthetic distributions seen during training.

Temporal Consistency. We evaluate temporal consistency from both a pixel-level perspective and an optical-flow consistency perspective. As shown in Tab. 1, DeCache achieves Ewarp and tOF scores comparable to or better than the baselines, indicating that our method maintains stable and coherent temporal behavior over long sequences without sacrificing frame-to-frame consistency from the added perceptual enhancement.

Qualitative Results. Fig. 2 presents visual comparisons. DeCache effectively handles complex real-world degradations, producing sharper and more realistic results. For example, in the first sample, the input suffers from severe ghosting artifacts caused by motion, and FlashVSR still exhibits residual ghosting and blurring around moving objects (e.g., the foreground pole), whereas our method successfully eliminates these ghosting artifacts and recovers sharp, well-defined structures. More visual results are provided in the supplementary material.

Table 1: **Quantitative comparison of our method against state-of-the-art methods across multiple datasets.** The best and second best results are highlighted.

Dataset	Metric	STAR	DOVE	SeedVR2	FlashVSR	InstantViR	Ours
SRWOV/real	NIQE↓	5.415	5.622	5.291	5.000	5.882	4.843
	MUSIQ↑	46.86	58.04	56.91	63.15	53.66	64.19
	CLIP-IQA↑	0.3457	0.5166	0.5221	0.6503	0.4577	0.6555
	FasterVQA↑	0.4072	0.5397	0.4419	0.5544	0.3976	0.5613
	DOVER↑	-3.64	-2.20	-2.47	1.38	-5.79	1.94
AbsoluteDegradation (real)	NIQE↓	5.256	4.607	4.535	3.275	6.429	3.149
	MUSIQ↑	38.71	45.24	43.68	59.83	32.74	62.15
	CLIP-IQA↑	0.2734	0.3379	0.3300	0.5391	0.2370	0.5718
	FasterVQA↑	0.3101	0.4455	0.3060	0.6382	0.1676	0.6647
	DOVER↑	-3.85	-3.99	-5.28	-2.09	-9.89	-0.47
BSCV	PSNR↑	23.82	27.05	25.10	23.11	26.95	22.59
	SSIM↑	0.7487	0.8437	0.7840	0.6966	0.8318	0.6843
	LPIPS↓	0.2594	0.1855	0.2687	0.3039	0.1766	0.2995
	NIQE↓	5.153	5.200	4.571	3.696	5.314	3.518
	MUSIQ↑	55.34	54.55	58.10	67.00	51.41	68.13
	CLIP-IQA↑	0.3162	0.3547	0.4017	0.5574	0.3135	0.5911
	FasterVQA↑	0.8034	0.7869	0.7864	0.8391	0.6859	0.8393
	DOVER↑	9.94	8.60	9.82	10.77	6.16	11.92
	E_{warp}↓	4.825	2.266	4.113	4.389	4.03	3.963
	tOF↓	2.145	1.828	2.123	2.821	1.471	2.792
TAPE	PSNR↑	19.70	21.24	20.51	20.24	20.89	20.08
	SSIM↑	0.6021	0.6737	0.6338	0.6248	0.6284	0.6343
	LPIPS↓	0.4373	0.3137	0.3546	0.2706	0.3706	0.2799
	NIQE↓	4.615	6.154	4.353	4.089	5.243	4.145
	MUSIQ↑	44.40	52.61	54.69	70.91	48.90	72.28
	CLIP-IQA↑	0.2806	0.3394	0.3522	0.5848	0.3407	0.5995
	FasterVQA↑	0.6068	0.6587	0.6002	0.7908	–	0.7940
	DOVER↑	2.48	5.27	3.82	8.98	0.44	9.32
	E_{warp}↓	3.33	1.22	5.31	3.52	5.22	3.31
	tOF↓	2.106	0.992	2.826	1.476	3.919	1.457
AbsoluteDegradation (synthetic)	PSNR↑	18.59	19.17	17.95	17.72	18.99	17.77
	SSIM↑	0.5209	0.5566	0.4958	0.4772	0.5477	0.4791
	LPIPS↓	0.4635	0.3706	0.3395	0.3622	0.4196	0.3605
	NIQE↓	4.968	4.173	3.280	3.108	5.984	3.099
	MUSIQ↑	40.31	49.24	58.33	60.73	37.41	61.73
	CLIP-IQA↑	0.2043	0.2397	0.3843	0.4495	0.1948	0.4728
	FasterVQA↑	0.3528	0.5275	0.5653	0.5565	0.2127	0.5760
	DOVER↑	-4.29	-3.00	-1.23	-1.04	-7.16	0.09
	E_{warp}↓	21.65	14.22	22.36	20.87	20.96	20.49
	tOF↓	3.817	3.763	3.853	4.846	3.456	4.838
BSD	PSNR↑	21.92	24.60	24.07	21.63	25.13	21.69
	SSIM↑	0.6763	0.7644	0.7365	0.6551	0.7583	0.6572
	LPIPS↓	0.3552	0.3112	0.2786	0.3285	0.2834	0.3306
	NIQE↓	5.639	6.118	5.537	4.938	6.347	4.994
	MUSIQ↑	36.69	39.46	41.98	56.97	36.54	58.53
	CLIP-IQA↑	0.2184	0.2248	0.2035	0.3811	0.1938	0.4057
	FasterVQA↑	0.5854	0.5962	0.6161	0.6207	0.5035	0.6213
	DOVER↑	0.25	-0.21	0.74	1.74	-1.64	2.17
	E_{warp}↓	9.69	8.75	7.28	7.46	3.92	5.26
	tOF↓	7.063	6.957	6.006	9.035	2.967	8.957

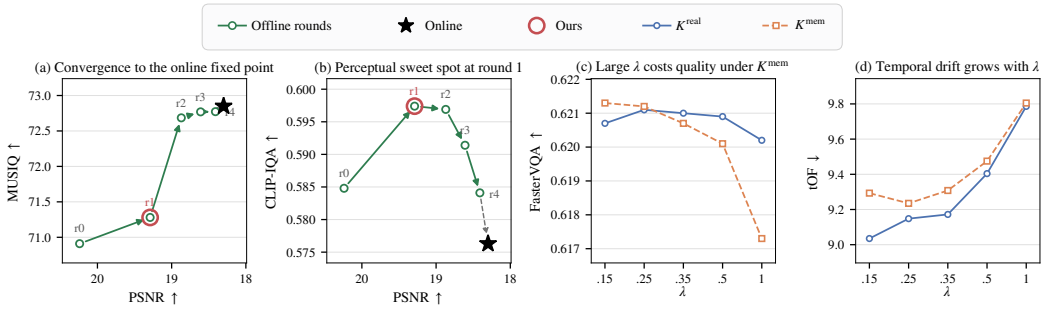

Figure 3: **Ablations. (a,b)** Round ablation on TAPE. **(a)** PSNR and MUSIQ move monotonically from round 0 toward the recursive online schedule, consistent with the offline Jacobi iteration and the online schedule converging to the same fixed point. **(b)** The aesthetic-oriented CLIP-IQA score peaks at round 1 and declines through rounds 2–3 toward the online value, marking a perceptual sweet spot. **(c,d)** Joint ablation of the value-mixing weight λ and key source (K^{real} vs. K^{mem}) on BSD, one refinement round.

Efficiency. DECACHE is training-free and plug-and-play. It costs a fixed constant factor per streaming step (three backbone forwards and two decoder invocations against one of each), giving $0.45\times$ the baseline throughput at an unchanged lookahead latency, and matching the cost of the offline two-pass realisation of the same transformation ($0.43\times$) while requiring only a single causal pass. Peak memory is 9.8 vs. 8.0 GB on 173-frame clips, but grows $3.5\times$ more slowly with sequence length (1.55 vs. 5.38 MB/frame), so it falls below the baseline on long sequences. See Appendix A.

4.3 ABLATION STUDY

Our ablation study examines three factors: the number of refinement rounds, the value-mixing weight λ, and the source of cached keys.

Why it_1 and Not the Fixed Point. Fig. 3 (a,b) sweeps the offline Jacobi iteration from round 0 (unmodified backbone) to round 4 ($Y^{(4)}$) and compares it against the recursive online schedule. Full-reference fidelity decreases monotonically with each round, consistent with the offline and online schedules converging toward the same attractor. Aesthetic-oriented metrics instead reach their best value at it_1. We adopt it_1 as our default, which captures most of the perceptual gain at a fraction of the fidelity cost of further iteration.

Value-mixing weight and key source. Fig. 3 (c,d) sweeps $\lambda \in \{0.15, 0.25, 0.35, 0.5, 1\}$ together with the source of cached keys, K^{real} vs. K^{mem}, on BSD, with the refinement round fixed at 1. Final configurations are reported in Appendix B.1.

5 CONCLUSION

We identified causal cache contamination as a failure mode of streaming video restoration: because a single forward pass both renders a chunk and writes its key–value states, degraded observations persist in memory after the input itself has recovered. DeCache addresses this without training by conditioning rendering on the original observation and memory writing on a pseudo-clean estimate, with damped value feedback trading artifact suppression against fidelity. We further showed that the online and offline schedules share a unique fixed point, and found empirically that iterating toward it reduces fidelity, whereas a single draft-seeded round, realized exactly in streaming with a one-chunk lookahead, captures most of the perceptual gain. Across benchmarks with severe synthetic and real-world degradations, DeCache improves perceptual quality over its FlashVSR backbone while keeping fidelity and temporal consistency competitive.

AI USE STATEMENT

In this work, we did not use generative AI tools for research ideation, experimental design, methodology development, or writing the core content of this paper. We used generative AI tools only for minor grammar and language polishing of the manuscript text. We have reviewed all AI-assisted text to ensure it accurately reflects our original intent and findings. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and agree to abide by the ICLR Code of Ethics and Code of Conduct. This work does not involve sensitive personal data or experiments with human subjects. Our experiments are conducted on publicly available datasets that are widely adopted in prior research and do not contain personally identifiable information (see Appendix B.2 for the full list). We adhere to the ICLR Code of Ethics regarding fairness, integrity, and responsible use of data and models, and we do not foresee any ethical concerns raised by the proposed method. All claims and findings in this paper are reported honestly and transparently.

REPRODUCIBILITY STATEMENT

DeCache is training-free and introduces no learned parameters; it is applied to the publicly released pretrained FlashVSR checkpoint (Zhuang et al., 2026). The method is fully specified in Sec. 3, with pseudocode for the offline and streaming variants in Algorithm 1 and Algorithm 2. Proofs of the convergence and schedule-equivalence results are given in Appendix C. Dataset versions, splits, frame counts, and preprocessing are described in Appendix B.2. Evaluation uses publicly available implementations of all reported metrics. Code is provided in the supplementary material.

REFERENCES

Lorenzo Agnolucci, Leonardo Galteri, Marco Bertini, and Alberto Del Bimbo. Reference-based restoration of digitized analog videotapes. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1648–1657. IEEE, 2024.

Weimin Bai, Suzhe Xu, Yiwei Ren, Jinhua Hao, Ming Sun, Wenzheng Chen, and He Sun. Instantvir: Real-time video inverse problem solver with distilled diffusion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16583–16592, 2026.

Yochai Blau and Tomer Michaeli. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6228–6237. IEEE, 2018.

Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4945–4954. IEEE, 2021.

Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Basicvsr++: Improving video super-resolution with enhanced propagation and alignment. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5962–5971. IEEE, 2022a.

Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Investigating tradeoffs in real-world video super-resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5952–5961. IEEE, 2022b.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Zheng Chen, Zichen Zou, Kewei Zhang, Xiongfei Su, Xin Yuan, Yong Guo, and Yulun Zhang. Dove: Efficient one-step diffusion model for real-world video super-resolution. *Advances in Neural Information Processing Systems*, 38:85218–85237, 2026.

Benjamin Naoto Chiche, Arnaud Woiselle, Joana Frontera-Pons, and Jean-Luc Starck. Stable long-term recurrent video super-resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 827–836. IEEE, 2022.

Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020a.

Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020b.

Peng Dai, Xin Yu, Lan Ma, Baoheng Zhang, Jia Li, Wenbo Li, Jiajun Shen, and Xiaojuan Qi. Video demoireing with relation-based temporal consistency. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17601–17610. IEEE, 2022.

Dror Freirich, Tomer Michaeli, and Ron Meir. A theory of the distortion-perception tradeoff in wasserstein space. *Advances in Neural Information Processing Systems*, 34:25661–25672, 2021.

Dario Fuoli, Martin Danelljan, Radu Timofte, and Luc Van Gool. Fast online video super-resolution with deformable attention pyramid. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1735–1744. IEEE, 2023.

Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, Jun Xiao, and Long Chen. Ca2-vdm: Efficient autoregressive video diffusion model with causal generation and cache sharing. *arXiv preprint arXiv:2411.16375*, 2024.

Shi Guo, Jianqi Ma, Xi Yang, Zhengqiang Zhang, and Lei Zhang. Toward accurate and temporally consistent video restoration from raw data. *arXiv preprint arXiv:2312.16247*, 2023.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

Mikolaj Jastrzebski, Dawid Glinkowski, Dawid Zieliński, Daniel Borkowski, Wojciech Kozłowski, and Kamil Adamczewski. Absolutedegradation: A physics-inspired synthetic film-degradation pipeline and archival film restoration benchmark. *arXiv preprint arXiv:2607.02131*, 2026.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5128–5137. IEEE, 2021.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European conference on computer vision*, pp. 179–195. Springer, 2018.

Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc V Gool. Recurrent video restoration transformer with guided deformable attention. *Advances in neural information processing systems*, 35:378–393, 2022.

Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *IEEE Transactions on Image Processing*, 33:2171–2182, 2024.

Tianyi Liu, Kejun Wu, Yi Wang, Wenyang Liu, Kim-Hui Yap, and Lap-Pui Chau. Bitstream-corrupted video recovery: A novel benchmark dataset and method. *Advances in Neural Information Processing Systems*, 36:68420–68433, 2023.

Yudong Mao, Hao Luo, Zhiwei Zhong, Peilin Chen, Zhijiang Zhang, and Shiqi Wang. Making old film great again: Degradation-aware state space model for old film restoration. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28039–28049. IEEE, 2025.

Matiur Rahman Minar, Seunghun Oh, GangHyeon Jeong, and Unsang Park. Steady-forcing: Balancing spatial persistence and motion continuity in long-horizon nature video diffusion. *arXiv preprint arXiv:2606.14732*, 2026.

Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

Xinmin Qiu, Congying Han, Zicheng Zhang, Bonan Li, Tiande Guo, Pingyu Wang, and Xuecheng Nie. Blazebvd: Make scale-time equalization great again for blind video deflickering. In *European Conference on Computer Vision*, pp. 37–53. Springer, 2024.

Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6626–6634. IEEE, 2018.

Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 2555–2563, 2023.

Jianyi Wang, Shanchuan Lin, Zhijie Lin, Yuxi Ren, Meng Wei, Zongsheng Yue, Shangchen Zhou, Hao Chen, Yang Zhao, Ceyuan Yang, et al. Seedvr2: One-step video restoration via diffusion adversarial post-training. In *International Conference on Learning Representations*, volume 2026, pp. 40987–41009, 2026.

Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1954–1963. IEEE, 2019.

H Wu. Open source deep end-to-end video quality assessment toolbox. *URL: http://github.com/timothyhtimothy/fast-vqa*, 2022.

Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring video quality assessment on user generated contents from aesthetic and technical perspectives. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20087–20097. IEEE, 2023.

Liangbin Xie, Xintao Wang, Shuwei Shi, Jinjin Gu, Chao Dong, and Ying Shan. Mitigating artifacts in real-world video super-resolution models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2956–2964, 2023.

Rui Xie, Yinhong Liu, Penghao Zhou, Chen Zhao, Jun Zhou, Kai Zhang, Zhenyu Zhang, Jian Yang, Zhenheng Yang, and Ying Tai. Star: Spatial-temporal augmentation with text-to-video models for real-world video super-resolution. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17108–17118. IEEE, 2025.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025.

Xi Yang, Chenhang He, Jianqi Ma, and Lei Zhang. Motion-guided latent diffusion for temporally consistent real-world video super-resolution. In *European conference on computer vision*, pp. 224–242. Springer, 2024.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974. IEEE, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Ziqing Zhang, Kai Liu, Zheng Chen, Xi Li, Yucong Chen, Bingnan Duan, Linghe Kong, and Yulun Zhang. Infvsr: Toward consistency-driven streaming generative video super-resolution. *arXiv preprint arXiv:2510.00948*, 2025.

Zhihang Zhong, Ye Gao, Yinqiang Zheng, Bo Zheng, and Imari Sato. Real-world video deblurring: A benchmark dataset and an efficient recurrent neural network. *International Journal of Computer Vision*, 131(1):284–301, 2023.

Shangchen Zhou, Peiqing Yang, Jianyi Wang, Yihang Luo, and Chen Change Loy. Upscale-a-video: Temporal-consistent diffusion model for real-world video super-resolution. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2535–2545. IEEE, 2024.

Junhao Zhuang, Shi Guo, Xin Cai, Xiaohui Li, Yihao Liu, Chun Yuan, and Tianfan Xue. Flashvsr: Towards real-time diffusion-based streaming video super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43482–43493, 2026.

A Implementation Details

We implement streaming DeCache by interleaving reference reconstruction, memory purification, and rendering. The reference condition is generated and consumed incrementally, while each rendering step receives the same refined history as in the offline computation. This section presents the schedule and its equivalence argument, followed by resource measurements. Algorithm 2 summarizes the implementation.

A.1 Causal Execution Schedule

Temporal alignment. We use zero-based frame and step indices, with T streaming steps. Let I_t and J_t be the LR input and decoded-output frame intervals at step t, respectively. For the evaluated backbone,

$$\begin{aligned} I_0 &= [0, 24], & J_0 &= [0, 20], \\ I_t &= [8t + 17, 8t + 24], & J_t &= [8t + 13, 8t + 20] \quad (t \geq 1). \end{aligned}$$

All intervals are inclusive. Let $\mathcal{C}_t$ denote the refined KV state after writing step t, with eviction handled by the backbone's fixed FIFO window, and let $\mathcal{C}_{-1}$ be empty. The memory condition $\tilde{x}$ is constructed from the independent reference reconstruction.

After reference step t, this condition is available through frame $8t + 20$. Memory construction for step $t - 1$ requires frames only through $8(t - 1) + 24 = 8t + 16$. We therefore execute

$$\mathrm{REF}(t) \longrightarrow \mathrm{MEM}(t - 1) \longrightarrow \mathrm{OUT}(t).$$

The reference computation produces the next seed chunk; $\mathrm{MEM}(t - 1)$ reads $\mathcal{C}_{t-2}$ and commits $\mathcal{C}_{t-1}$; and $\mathrm{OUT}(t)$ renders from the real LR input and $\mathcal{C}_{t-1}$. At $t = 0$, the memory operation is omitted. At the end of a finite clip, the last memory write can also be omitted because no later output consumes it. The implementation thus performs $3T - 1$ DiT evaluations and $2T$ incremental decoder calls.

Preserving the input-frame requirements. The reference and rendering computations at iteration t consume the same LR window. The schedule therefore requires no additional LR-frame lookahead beyond the backbone's chunk dependencies. The memory write is delayed in execution order until its reference condition is available; it is still completed before the rendering computation that needs it. The additional processing cost is captured by the runtime measurements in Appendix A.3.

Algorithm notation. In Algorithm 2, $\boldsymbol{z}_t$ denotes the fixed noise slice for step t, reused by its reference, rendering, and memory evaluations. $\mathcal{D}_{\mathrm{base}}$ and $\mathcal{D}_{\mathrm{out}}$ denote incremental reference and rendering decoding, including the same framewise output postprocessing as offline. The causal states are maintained separately as described in Appendix A.2. Eval_t evaluates F at step t and returns its prediction and complete candidate KV stores after the backbone's capacity-k FIFO update. It leaves the incoming history unchanged until an explicit assignment commits a candidate; returned stores are therefore assigned directly, without a second append or FIFO operation.

Equivalence to offline refinement. Consider offline and interleaved executions with identical weights, per-step noise, real LR and seed tensors, position indices, initialized projection and decoder states, FIFO rules, and numerical operations. Both executions maintain separate causal states for the logical streams while sharing their weights. The reference stream then reproduces the offline reference condition. Equivalence of the refinement computation follows by induction. Step zero starts with the same empty cache. If the incoming cache for step s agrees, the real-condition and seed-condition computations produce the same rendering output and cache candidates in both executions. Applying the same key selection and value mixture yields the same $\mathcal{C}_s$. Moving its construction from after offline output s to before interleaved output $s + 1$ preserves every input to that operation. The output latents and their corresponding decoder histories consequently agree.

Damped values and key selection. In the algorithm, q selects the key source and λ denotes the value-mixing coefficient λ_V reported in the experiments. For real and memory candidates A and B,

Algorithm 2 Streaming DeCache

Require: LR stream $\boldsymbol{x}$; DiT F; projections $P_{\text{base}}, P, \tilde{P}$
Require: Number of streaming steps T; cache capacity k; mix $\lambda \in [0,1]$; $q \in \{\text{real}, \text{mem}\}$
Sample fixed noise $\boldsymbol{z}$; $\mathcal{C}_{\text{base}}, \mathcal{C} \leftarrow \varnothing$
$\tilde{X} \leftarrow \varnothing$; $(\bar{K}, \bar{V}) \leftarrow (\varnothing, \varnothing)$ ▷ rolling seed; pending real candidates
Initialize independent projection and decoder states
for $t = 0$ **to** $T-1$ **do**
  $(\boldsymbol{v}_t^b, K^b, V^b) \leftarrow \text{Eval}_t(\boldsymbol{z}_t; P_{\text{base}}(\boldsymbol{x}[I_t]), \mathcal{C}_{\text{base}})$ ▷ $\text{REF}(t)$
  $\mathcal{C}_{\text{base}} \leftarrow (K^b, V^b)$
  $\boldsymbol{y}_t^b \leftarrow \mathcal{D}_{\text{base}}(\boldsymbol{z}_t - \boldsymbol{v}_t^b)$
  Append $\mathcal{B}(\boldsymbol{y}_t^b, \boldsymbol{x}[J_t])$ to $\tilde{X}$ ▷ same seed construction as offline
  if $t > 0$ **then**
    $s \leftarrow t-1$
    $(\cdot, K^m, V^m) \leftarrow \text{Eval}_s(\boldsymbol{z}_s; \tilde{P}(\tilde{X}[I_s]), \mathcal{C})$ ▷ $\text{MEM}(t-1)$
    $K_s \leftarrow \text{Select}_q(\bar{K}, K^m)$; $V_s \leftarrow \text{Mix}_\lambda(\bar{V}, V^m)$
    $\mathcal{C} \leftarrow (K_s, V_s)$ ▷ commit before rendering step t
    $(\bar{K}, \bar{V}) \leftarrow (\varnothing, \varnothing)$
    Release seed chunks preceding the next memory window
  end if
  $(\boldsymbol{v}_t, K^r, V^r) \leftarrow \text{Eval}_t(\boldsymbol{z}_t; P(\boldsymbol{x}[I_t]), \mathcal{C})$ ▷ $\text{OUT}(t)$; leave $\mathcal{C}$ unchanged
  if $t < T-1$ **and** $(q = \text{real}$ **or** $\lambda < 1)$ **then**
    $(\bar{K}, \bar{V}) \leftarrow (K^r, V^r)$ ▷ retain for $\text{MEM}(t)$ next iteration
  end if
  emit $\mathcal{D}_{\text{out}}(\boldsymbol{z}_t - \boldsymbol{v}_t)$
end for

respectively, the helper operations are

$$\text{Select}_q(A,B) = \begin{cases} A, & q = \text{real}, \\ B, & q = \text{mem}, \end{cases} \qquad \text{Mix}_\lambda(A,B) = \begin{cases} A, & \lambda = 0, \\ (1-\lambda)A + \lambda B, & 0 < \lambda < 1, \\ B, & \lambda = 1. \end{cases}$$

These operations act blockwise on the complete candidate stores. The real candidates $(\bar{K}, \bar{V})$ from $\text{OUT}(s)$ are retained until $\text{MEM}(s)$ executes at iteration $s+1$. Both candidates are computed from the same incoming history $\mathcal{C}_{s-1}$, so the implementation applies the same whole-tensor mixture as the offline version. Selecting real keys uses the retained rendering-pass keys. The candidates are released after the write. For $q = \text{mem}$ and $\lambda = 1$, both helpers select the memory candidates directly, so the unused real candidates may remain empty.

A.2 Shared Weights and Bounded State

The reference, memory, and rendering computations share one frozen set of model weights. Each conditioning stream keeps its own convolutional state, and the reference and rendering decoder streams keep independent causal states. The reference and refinement KV stores have fixed FIFO capacity. When required, the rendering cache candidates are retained for one iteration, and seed chunks are released after their last use. These recurrent states and the seed window have bounded size with respect to video length.

We implement this schedule with four memory optimizations: directly reusing the rolled cache returned by the backbone; sharing conditioning-projection weights while swapping stream states; materializing cache slices to release unused parent storage; and keeping a rolling seed window. Together they reduce peak allocated memory from 14.68 to 9.81 GiB at 512×512, and from 36.24 to 23.29 GiB at 1280×640 (Tab. 2).

For the file-based benchmark, the loader preloads the upsampled LR input and latent noise, and output images accumulate in host memory. These full-clip buffers are distinct from the bounded recurrent state. At 512×512, the bf16 input and noise tensors contribute approximately 1.5 and

Table 2: Peak allocated GPU memory (GiB) across implementation stages, for 180 input frames yielding 173 outputs. All rows use $K = K^{\mathrm{mem}}, \lambda_V = 1$ and six timed runs per resolution.

Implementation	512×512	1280×640
Initial	14.68	36.24
Cache handling and shared projections	10.06	24.06
Including a rolling seed window	9.81	23.29

Table 3: Throughput and median peak GPU memory on 180 input / 173 output frames. Refinement rows use $K = K^{\mathrm{mem}}, \lambda_V = 1$. The offline refinement stage consumes a precomputed seed. Alloc. and Resv. denote tensor allocation and caching-allocator reservation, respectively, in GiB.

	512×512			1280×640		
Method	fps	Alloc.	Resv.	fps	Alloc.	Resv.
FlashVSR	20.24	8.00	9.96	6.56	17.64	23.60
Offline refinement stage	15.11	8.81	11.22	4.84	19.02	26.48
DECACHE, interleaved	9.10	9.81	12.25	2.88	23.29	30.48

0.03125 MiB per stored frame, respectively, accounting for most of the remaining length-dependent GPU allocation. Their measured effect is reported in Appendix A.4.

A.3 EFFICIENCY PROTOCOL AND RESULTS

Protocol. We benchmark the full memory-replacement setting ($K = K^{\mathrm{mem}}, \lambda_V = 1$) on one NVIDIA A800-80GB, using bf16 model computations, batch size one, one denoising step, and no VAE tiling. We fix seed zero and set `sparse-ratio=2`, `kv-ratio=3`, and `local-range=11`. For each method and dataset, we report the median of six runs: two timed repetitions per clip following one discarded warm-up. We use TAPE clips 000, 013, and 016, and BSCV clips 00bd64cb00, 012257ffcf, and 01475d1fe7. Temporal preprocessing maps each 180-frame input to 173 output frames, the numerator used for FPS.

Methods run in separate processes, with model loading excluded from timing and CUDA synchronization bracketing each measurement. Timing includes input loading and preprocessing, streaming reference and seed construction, inference, decoding, color correction, and conversion to host-resident images; disk writes are excluded. Per-run peak allocated and reserved GPU memory includes model weights. Baseline and offline methods decode after chunk processing, while the interleaved implementation decodes incrementally.

Throughput and memory. Table 3 reports the measurements. At 512×512 and 1280×640, the complete interleaved implementation runs at 9.10 and 2.88 fps, corresponding to $0.45\times$ and $0.44\times$ baseline throughput. Its median wall times are 19.00 and 60.02 s, compared with 8.55 and 26.36 s for the baseline. Thus the reference, purification, and rendering computations together take approximately $2.22\times$ and $2.28\times$ the baseline time while sharing the same model weights.

Cost of the offline realization. Offline refinement additionally requires a complete reference run. Summing the median reference and refinement times gives stage-sum estimates of 19.9968 s (8.65 fps) and 62.1013 s (2.79 fps) at the two resolutions. These estimates exclude seed construction and intermediate output serialization. The measured interleaved execution has a comparable runtime scale while constructing and consuming its reference condition incrementally.

A.4 MEMORY SCALING WITH CLIP LENGTH

We also measure prefixes at four lengths (Tab. 4). The interleaved implementation's peak allocated memory grows from 9.69 to 10.65 GiB between 100 and 733 input frames. Over the same interval, the evaluated baseline and offline refinement paths grow from 7.76 to 11.08 GiB and from 8.45 to 12.71 GiB, respectively. The additional reference state contributes a fixed cost, while incremental decoding and seed buffering reduce the growth observed in these implementations.

Table 4: Peak allocated GPU memory (GiB) on prefixes of one TAPE clip at 512×512. Both refinement paths use $K = K^{\mathrm{mem}}, \lambda_V = 1$. The offline row measures its refinement stage.

Input frames	100	200	400	733
Output frames	93	197	397	733
FlashVSR	7.76	8.08	8.70	11.08
Offline refinement stage	8.45	8.92	9.84	12.71
DECACHE, interleaved	9.69	9.85	10.15	10.65

The endpoint finite-difference growth rates over this interval are 5.38, 6.90, and 1.55 MiB per input frame for the baseline, offline refinement, and interleaved implementation, respectively. On the 733-frame clip, the interleaved implementation therefore uses less allocated GPU memory than both comparison paths. These rates summarize the measured interval and the stated buffering implementations.

A.5 STAGE PROFILING

Exploratory profiling attributes approximately 51–52% of the instrumented time to the two decoder streams combined, 38–39% to the forward-stage regions, and about 10% to reference postprocessing and seed construction. The percentages are normalized by the sum of the instrumented regions; input preparation and rendering postprocessing are outside that sum. The seed region includes color correction, image operations, and device transfers. This breakdown identifies reference decoding and seed construction as useful targets for further optimization, particularly because the decoded reference is subsequently downsampled to form the memory condition.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 HYPERPARAMETER SELECTION AND FINAL CONFIGURATIONS

We search over $K \in \{K^{\mathrm{real}}, K^{\mathrm{mem}}\}$ and $\lambda \in \{0.15, 0.25, 0.35, 0.5, 1.0\}$, with the refinement round fixed to one. We selected the configuration by jointly considering reconstruction fidelity, perceptual quality, and temporal consistency, seeking a favorable overall trade-off. See Tab. 5. Under MSE distortion and the Wasserstein-2 perception metric, (Freirich et al., 2021) show that optimal estimators along the perception–distortion trade-off can be constructed as convex combinations of an MMSE estimator and a minimum-MSE estimator subject to perfect perceptual quality. This interpolation principle motivates blending real-conditioned and pseudo-clean-conditioned values to regulate restoration-derived feedback. The theorem concerns reconstruction estimators; our application operates on internal memory features.

Table 5: Selection configurations across evaluation datasets.

Evaluation dataset	**Key source**	λ	**Refinement rounds**
TAPE	K^{real}	0.5	1
BSCV	K^{mem}	0.15	1
BSD	K^{mem}	0.15	1
AbsoluteDegradation (synthetic)	K^{mem}	0.15	1
AbsoluteDegradation (real)	K^{real}	1.0	1
SRWOV/real	K^{mem}	1.0	1

B.2 BENCHMARK CONSTRUCTION AND EVALUATION PREPROCESSING

We evaluate on TAPE, BSCV, BSD, AbsoluteDegradation, and SRWOV/real. TAPE, BSCV, AbsoluteDegradation (synthetic), and BSD provide ground-truth (GT) references, whereas AbsoluteDegradation (real) and SRWOV/real do not. Below, we describe the original construction of each benchmark and the data selection and preprocessing used in our experiments.

TAPE. TAPE (Agnolucci et al., 2024) is constructed from 4K video clips of the "Venice" scene in the Harmonic free demo footage dataset. The authors generate degraded videos through a synthetic degradation pipeline implemented in Adobe After Effects, covering five degradation types: (i) undersaturation, (ii) tape noise, (iii) tape dropout, (iv) tape mistracking / VHS edge waving, and (v) chroma loss / scanline flickering. These degradations are distributed over time according to independent, per-frame random sampling of artifact combination, spatial position, and intensity, so that no temporal consistency is enforced across consecutive frames, mimicking the abrupt frame-to-frame variation observed in real archival footage. The corresponding GT videos are the original clean frames prior to synthetic degradation. We use all 40 clips from both the official training (30 clips) and test (10 clips) splits, since our method is training-free and thus does not require withholding clips for training. The evaluated clips contain 26,392 frames in total at a source resolution of 512×512.

BSCV. BSCV (Liu et al., 2023) is constructed from real H.264 bitstream corruption rather than manually designed error masks. The authors first encode source video clips (sampled from YouTube-VOS and DAVIS) into H.264 bitstreams using a closed-GOP configuration with a GOP size of 16 frames. They generate degraded videos through a three-parameter corruption model (P, L, S) that randomly removes segments of the frame-data NALUs in the bitstream. The corrupted bitstream is then parsed by a standard H.264 decoder to yield the final degraded video. This process naturally covers six degradation types: blocking artifacts, color artifacts, duplication artifacts, misalignment, texture loss, and trailing artifacts. These degradations are distributed over time according to the GOP structure and H.264's inter-frame prediction dependencies: since a decoding error in one frame propagates to subsequent frames that reference it until the next intra-refresh, corruption severity varies unpredictably and non-uniformly across a clip, rather than being independently sampled per frame. The corresponding GT videos are obtained by decoding the same source video from its original, corruption-free bitstream. We use all 661 clips from the official test split (`test_144096`) of the $(1/16, 0.4, 4096)$ branch, combining the YouTube-VOS and DAVIS test subsets. The evaluated clips contain 84,286 frames in total at their native source resolutions.

SRWOV/real. The real-world subset of SRWOV (Mao et al., 2025) consists of naturally degraded old film footage and therefore provides no ground-truth reference. To construct it, the authors collect old videos spanning diverse content categories and capturing periods, resulting in heterogeneous native resolutions and frame rates due to the variety of historical capturing devices. From each collected source video, multiple non-overlapping 5-second clips are extracted while avoiding excessive scene repetition, yielding 216 clips drawn from 55 distinct source videos, for a total of 29,256 frames.

BSD. BSD (Zhong et al., 2023) is constructed from real, physically captured blurry/sharp video pairs, using a co-axial beam-splitter acquisition system with two synchronized, identically positioned cameras sharing a single objective lens to avoid inter-reflection and ensure precise spatial alignment. This process covers a single degradation type, i.e., real camera motion blur, at three blur-intensity configurations defined by (sharp exposure time, blurry exposure time) pairs: 1ms–8ms, 2ms–16ms, and 3ms–24ms. Since the blur arises from real camera and/or object motion captured live, its temporal and spatial pattern is naturally determined by scene content rather than by any explicit randomization scheme. The corresponding GT videos are captured simultaneously by the co-located camera C2. We use all 60 official test clips (20 clips $\times$ 3 blur-intensity settings). The evaluated clips contain 9,000 frames in total (3,000 frames per setting $\times$ 3 settings) at a source resolution of 640×480.

AbsoluteDegradation. We evaluate on two subsets of AbsoluteDegradation (Jastrzebski et al., 2026): a real-world subset and a synthetic subset. The real subset consists of naturally degraded archival footage curated from 30 public-domain films (1896–1918) held by the Library of Congress, split into single-scene clips via automatic scene detection followed by manual curation; we use the compact evaluation variant, comprising 13,252 frames, which provides no ground truth. The synthetic subset is constructed by degrading the complete REDS sharp validation split (30 clips, 100 frames each, 3,000 frames total) with a six-stage, physics-inspired synthesis pipeline that composes seven analog artifact families (optical blur, film grain, scratches, dust/texture overlays, negative-film distortions, camera gate-weave, and compression/photometric shifts) under a three-tier severity

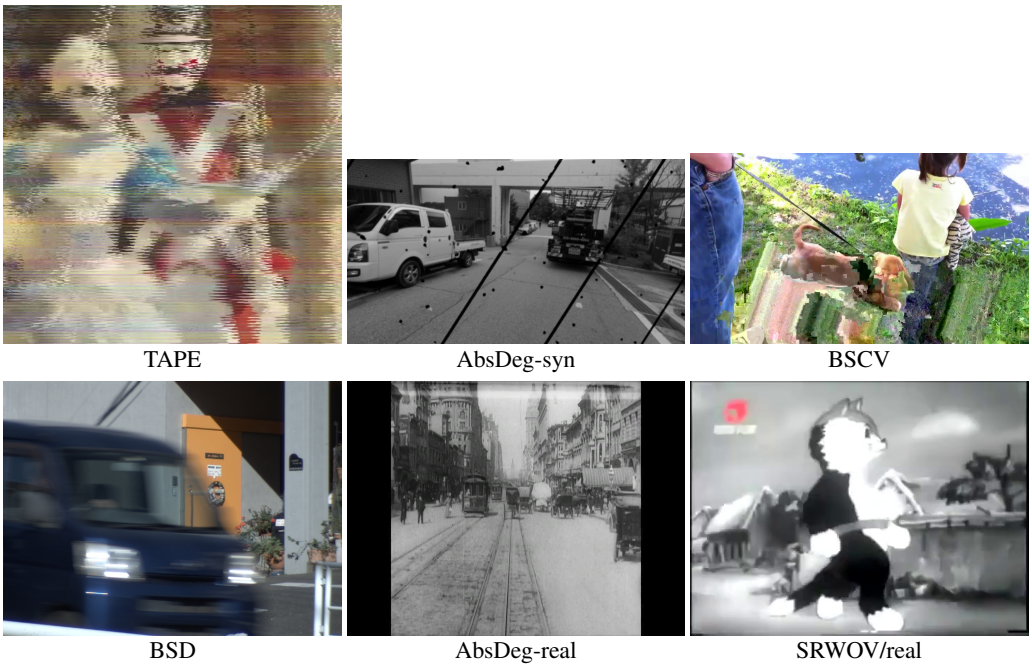

Figure 4: **Representative degraded inputs from the evaluated benchmarks.** Top row: synthetically generated degradations, including analog-tape artifacts (TAPE), physics-inspired film artifacts (AbsoluteDegradation-syn), and H.264 bitstream corruption (BSCV). Bottom row: real-world captured degradations, including camera motion blur (BSD, 3ms–24ms setting) and naturally aged archival films (AbsoluteDegradation-real, SRWOV/real). Frames are selected to illustrate typical severe cases.

curriculum (LIGHT/MEDIUM/HEAVY), with degradation parameters sampled per clip and held temporally coherent across frames; the corresponding clean REDS frames serve as pixel-aligned ground truth by construction.

Fig. 4 shows representative degraded frames from each benchmark, grouped into synthetically generated (top) and real-world captured (bottom) degradations. Together, they cover analog-tape artifacts, codec-level bitstream corruption, physical motion blur, and naturally aged archival film. In many cases, local content is severely corrupted or entirely missing, e.g., tape dropout in TAPE, lost macroblocks in BSCV, and scratches and dust in archival footage, making these benchmarks a stringent test of restoration robustness.

C Convergence and Optimality of the Relaxation Schedules

This appendix makes precise the claim of Sec. 3.4: the online and offline schedules solve the same self-consistency equation, this equation has a unique solution $Y^\star$ that both schedules reach, and $Y^\star$ is nonetheless the wrong target to run to convergence. Throughout, we fix a realization $\{\boldsymbol{z}_t\}_{t=1}^{T}$ of the per-chunk noise and treat M, Φ, $\downarrow_4$ as deterministic given this realization, so that the only remaining degree of freedom across schedules is what trajectory the memory path is seeded with. Recall

$$\big(\mathcal{M}(Y)\big)_t \;=\; M\Big(x_t;\, \Phi\big(\{\downarrow_4 (y_s)\}_{s\in W_t}\big)\Big), \qquad W_t \subseteq \{1,\dots,t-1\},\;\; |W_t| \le k. \tag{7}$$

The only structural fact we use is that W_t contains no index $\ge t$: the memory read by chunk t is assembled exclusively from chunks strictly before it.

C.1 THE CAUSAL FIXED POINT IS UNIQUE AND IS REACHED IN FINITELY MANY ROUNDS

Lemma 1 (Well-posedness). *There is exactly one trajectory $Y^\star = (y_1^\star, \dots, y_T^\star)$ with $Y^\star = \mathcal{M}(Y^\star)$, given by the forward recursion*

$$y_1^\star = M\big(x_1; \Phi(\varnothing)\big), \qquad y_t^\star = M\Big(x_t;\, \Phi\big(\{\downarrow_4 (y_s^\star)\}_{s\in W_t}\big)\Big) \;\; (t>1), \tag{8}$$

using already-determined $y_s^\star$, $s < t$.

Proof. Because $W_t \subseteq \{1, \dots, t-1\}$, the self-consistency equation $Y = \mathcal{M}(Y)$ is strictly lower-triangular in t: the t-th component of $\mathcal{M}(Y)$ never depends on $y_t, \dots, y_T$. Existence and uniqueness therefore follow by the same forward substitution used to solve a triangular linear system: y_1 is pinned down by $W_1 = \varnothing$ independently of the rest of Y; given $y_1, \dots, y_{t-1}$ pinned down, y_t is immediately pinned down as well. No index is ever left underdetermined or overdetermined. □

Proposition 1 (Finite Jacobi convergence). *For any seed $Y^{(0)}$ and $Y^{(n)} = \mathcal{M}(Y^{(n-1)})$, $n = 1, 2, \dots$,*

$$y_t^{(n)} = y_t^\star \qquad \text{for every } n \geq t. \tag{9}$$

In particular $Y^{(T)} = Y^\star$ exactly, after at most T rounds, regardless of the seed $Y^{(0)}$, the cache length k, or any regularity of M.

Proof. Induction on t. For $t = 1$: $W_1 = \varnothing$, so $(\mathcal{M}(Y))_1 = M(x_1; \Phi(\varnothing))$ for every Y; hence $y_1^{(n)} = y_1^\star$ for all $n \geq 1$. Assume the claim for all $s < t$. For $n \geq t$ we have $n - 1 \geq t - 1 \geq s$ for every $s \in W_t$, so by the inductive hypothesis $y_s^{(n-1)} = y_s^\star$ for all $s \in W_t$. Substituting, $y_t^{(n)} = M(x_t; \Phi(\{\downarrow_4 (y_s^\star)\}_{s\in W_t})) = y_t^\star$, which proves the claim for t. □

This is a stronger statement than a generic contraction argument: convergence is exact, in an a-priori known number of steps, and it needs no Lipschitz assumption on the denoiser M. This is a direct consequence of causality, exactly as a strictly-past dependency graph guarantees a well-defined, one-pass topological evaluation order. (The cache length k affects how much memory each step reads, but not this worst-case round count: even $W_t = \{1, \dots, t-1\}$ still needs t rounds in the worst case, since $y_{t-1}^\star$ itself may need $t - 1$ rounds to stabilize.)

C.2 ONLINE SINGLE-PASS IS THE OFFLINE FIXED POINT

Proposition 2 (Schedule equivalence). *Define the online trajectory Y^{online} by evaluating $y_t^{\text{online}} = M(x_t; \Phi(\{\downarrow_4 (y_s^{\text{online}})\}_{s\in W_t}))$ sequentially for $t = 1, \dots, T$, each chunk's memory drawing on already-decoded chunks of the very same sweep. Then, for the shared noise realization $\{\boldsymbol{z}_t\}$,*

$$Y^{\text{online}} = Y^\star. \tag{10}$$

Proof. Y^{online} is computed by exactly the forward recursion of Lemma 1: sequential evaluation in increasing t, each step consuming only already-fixed earlier outputs. By uniqueness (Lemma 1), $Y^{\text{online}} = Y^\star$. □

Propositions 1 and 2 together explain Fig. 3: online (Gauss–Seidel) and offline-at-convergence (Jacobi, it_∞) are not two methods that happen to agree, they are the same trajectory computed two different ways.

C.3 WHY THE FIXED POINT IS THE WRONG OPERATING POINT

$Y^\star$ satisfies $Y^\star = \mathcal{M}(Y^\star)$ purely because the rendering and memory paths agree with each other; no term in this condition references the true clean video y. Unrolling the recursion of Lemma 1 shows why this absence of grounding matters in practice. Chunk t's fixed-point memory is not one hop from real-LR evidence: every $y_s^\star$ feeding it, $s \in W_t$, is itself already a function of other regenerated, re-degraded outputs rather than of x_s directly (unless $s = 1$). By the time chunk t is reached, its memory has been synthesized and re-fed through M up to $t - 1$ times without ever reintroducing

new real-LR signal after the first chunk. Any systematic bias of M relative to the true posterior — the over-sharpening / added high-frequency detail that perceptually-trained restoration priors are known to exhibit relative to the MMSE-optimal reconstruction — therefore has no mechanism to be corrected along this chain: each relay can only compound the bias in the same direction, since the only input at each relay is the model's own prior belief about itself.

$Y^{(1)} = \mathrm{it}_1$ avoids this by construction: its memory is seeded once, directly, from $Y^{(0)}$, the vanilla trajectory computed from real-LR conditioning throughout (Eq. 2). Every memory value it reads is therefore at most one such relay away from real-LR grounding, bounding how much self-reinforcing bias can accumulate before the pass that actually renders the output. Fig. 6 and Fig. 3 make this monotone degradation from it_1 toward $Y^\star$ direct: fidelity to ground truth falls with each additional round even as the rounds converge (Proposition 1 guarantees they do converge, only not to a good place). Fig. 6 shows this pattern qualitatively at the recovery chunk $t^\star$ (Sec. 3.1): high-frequency detail is increasingly synthesized rather than recovered from it_2 onward, and by it_3 visibly diverges from the structure present.

We stress that this last argument, unlike Propositions 1 and 2, is not a theorem. Establishing it rigorously would require a quantitative bias assumption on M that we do not have independent access to. We therefore support it empirically rather than claim more than the causal structure gives us for free.

D THE PERCEPTION-DISTORTION TRADEOFF

Regression-oriented baselines are rewarded by full-reference metrics for being blurry. Streaming VSR under real degradations is an ill-posed inverse problem: a given low-resolution observation is consistent with many plausible high-resolution completions. As established by (Blau & Michaeli, 2018), the distortion-optimal estimator under this uncertainty is the posterior mean, and averaging over a distribution of plausible high-frequency completions necessarily attenuates high-frequency content, i.e., the MMSE-optimal reconstruction is blurry by construction, not by failure. DOVE and STAR are trained with an explicit emphasis on pixel-level reconstruction objectives (Chen et al., 2026; Xie et al., 2025), which pushes their outputs toward this regression-to-the-mean regime. Their higher PSNR/SSIM in Tab. 1 is therefore expected under the perception-distortion theory itself, rather than evidence of more faithful recovery of scene content: a method can score well on full-reference distortion precisely by suppressing the same high-frequency detail that no-reference perceptual metrics (MUSIQ (Ke et al., 2021), CLIP-IQA (Wang et al., 2023), DOVER (Wu et al., 2023)) and human viewing reward. Fig. 5 illustrates this directly: the DOVE/STAR columns are visibly smoother than DECACHE, particularly on fine texture and edges.

Blur is a confound in optical-flow-based temporal metrics. This same attenuation of high-frequency content also inflates E_{warp} (Lai et al., 2018) and tOF (Chu et al., 2020b). Both metrics score temporal consistency by warping adjacent frames with an estimated optical flow field and measuring the residual discrepancy. Prior work on warping-based temporal-consistency metrics has noted that they conflate genuine inconsistency with optical-flow estimation error, and that this estimation error is masked when the compared frames are blurred (Qiu et al., 2024), with related concerns motivating corrected variants of the metric (Guo et al., 2023; Dai et al., 2022). A blurrier sequence removes exactly the high-frequency structure whose mismatch such metrics would otherwise register, and warping-residual metrics are correspondingly less sensitive to any given registration error when little high-frequency content remains to reveal the mismatch. In other words, E_{warp}/tOF partially measure how much high-frequency content there is left to be inconsistent, not only how consistent it is. Consequently, a method that regresses toward the posterior mean receives a temporal-consistency score that is partly an artifact of its own blur, rather than a genuine advantage in modeling temporal coherence. Read against this confound, DECACHE's comparable-or-better E_{warp}/tOF in Tab. 1 is a stronger result than the raw numbers suggest: it maintains competitive temporal stability while retaining high-frequency detail that would otherwise be penalized by these metrics, rather than achieving it by discarding that detail.

Taken together, these two effects explain why full-reference fidelity and temporal-consistency metrics alone are insufficient to rank streaming VSR methods under the perception-distortion tradeoff.



Figure 5: **Qualitative comparison illustrating the perception–distortion trade-off.** On both fine, unstructured texture and structured edges, DOVE and STAR, trained with an explicit emphasis on pixel-level distortion, over-smooth high-frequency detail, while DECACHE recovers sharp silhouettes and well-defined contours. Despite the visibly higher perceptual fidelity of our results, DOVE and STAR still attain higher PSNR/SSIM on these benchmarks (Tab. 1): full-reference distortion metrics reward the smoother, lower-variance reconstruction over the perceptually faithful one.

E RESULTS ON STANDARD VSR BENCHMARKS

DECACHE is designed for inputs whose degradation is severe enough to corrupt the streaming memory. A natural question is whether it costs anything when the degradation is mild and well-modeled. We evaluate on four standard VSR benchmarks: REDS4, REDS30, SPMCS, and UDM10.

Setup. For REDS4 and REDS30 we use the official $\times 4$ bicubic LR frames. For SPMCS and UDM10 we use the LR frames distributed with each dataset. DECACHE uses a single fixed configuration on all four datasets ($\lambda = 0.15$, keys from the memory pass), with no per-dataset tuning. We report PSNR, SSIM, and LPIPS for fidelity, NIQE, MUSIQ, CLIP-IQA, FasterVQA, and the DOVER technical score for perceptual quality, E_{warp}, and tOF for temporal consistency. We omit the DOVER aesthetic score: SPMCS and UDM10 clips produce only 29 output frames, fewer than the 32 temporal fragments of DOVER's aesthetic branch, so its sampler degenerates to scoring the first frame only.

Results. Tab. 6 shows that DECACHE performs on par with or better than FlashVSR on most metrics across all four datasets. Together with the gains on corrupted inputs in Tab. 1, this shows that DECACHE's benefit comes at no cost when the input is not corrupted.

F SCOPE OF APPLICABILITY AND BACKBONE SELECTION

DECACHE is training-free: it does not modify weights, and instead intervenes on state that a backbone already maintains at inference time. Its applicability is therefore a structural property of the backbone rather than an empirical question, and we state the two conditions it relies on.

Table 6: Comparison on standard VSR benchmarks with mild degradation. ↑: higher is better; ↓: lower is better.

Dataset	Method	Fidelity			Perceptual					Temporal	
		PSNR↑	SSIM↑	LPIPS↓	NIQE↓	MUSIQ↑	CLIP-IQA↑	FasterVQA↑	DOVERtec↑	tOF↓	E_{warp} ↓
REDS4	FlashVSR	21.67	0.5932	0.2535	2.745	66.65	0.4057	0.7856	−9.042	2.138	5.349
	DECACHE	21.69	0.5928	0.2549	2.737	66.86	0.4073	0.7855	−8.9799	2.157	5.286
REDS30	FlashVSR	21.59	0.5599	0.2599	2.691	65.23	0.4201	0.7481	−9.1112	1.767	5.954
	DECACHE	21.59	0.5596	0.2615	2.693	65.36	0.4217	0.7493	−9.1083	1.810	5.917
SPMCS	FlashVSR	21.26	0.5418	0.2767	3.395	71.61	0.5653	0.7845	−6.357	0.289	0.651
	DECACHE	21.27	0.5421	0.2767	3.394	71.62	0.5657	0.7846	−6.357	0.288	0.652
UDM10	FlashVSR	23.68	0.6966	0.2806	3.985	68.68	0.5553	0.8323	−5.178	0.823	0.946
	DECACHE	23.70	0.6977	0.2808	3.9853	68.73	0.5577	0.8325	−5.137	0.823	0.947

Assumption A1 (managed streaming cache). The backbone performs causal inference over an unbounded sequence with a KV cache whose contents must be managed at run time (evicted, rolled, or re-anchored) because the cache budget is smaller than the sequence.

Assumption A2 (persistent LQ conditioning). The degraded observation enters through a conditioning pathway that persists in attention (e.g. LQ-derived keys and values), rather than only as the initial state of the denoising trajectory.

A1 supplies the state our cache-level operator acts on; A2 supplies the measurement signal our reweighting acts on. Without A1 there is no cache pressure for a training-free policy to resolve and the operator reduces to the identity; without A2 no LQ-derived tensor is exposed to act on. We therefore scope our claims to streaming VSR systems satisfying both conditions. Table 7 places recent diffusion VSR systems in this design space.

Table 7: Design space of recent diffusion VSR systems. Only systems satisfying both A1 and A2 admit training-free cache-level intervention.

Method	Sequence horizon	Cache management	LQ conditioning	In scope
STAR / DOVE / SeedVR2	chunked, bidirectional	—	persistent	✗
InstantViR	bounded (81 frames)	—	initialisation only	✗
FlashVSR	unbounded	rolling	persistent	✓
InfVSR[†]	unbounded	rolling	persistent	✓

[†] In scope by architecture, but weights and code are not publicly available; see below.

Systems outside the class. A natural question is whether causal, KV-cached video restoration models admit our intervention in general. They do not, and the distinction our assumptions capture is between models that are block-causal within a fixed-length clip and models that are streaming. InstantViR (Bai et al., 2026) is representative of the former: it is a unified latent-space inverse-problem solver (super-resolution, deblurring and inpainting under one model) designed for clips of a fixed length, with a correspondingly fixed cache horizon, and it conditions on the degraded observation by using it to initialise the denoising trajectory. Both choices are well matched to that problem setting, and are simply orthogonal to A1 and A2: applying our method there would not probe generality across streaming KV architectures, but a different architectural class.

In-class systems we could not evaluate. InfVSR (Zhang et al., 2025) is, by its own description, the closest published system to the class we target: it adapts a pretrained DiT into a causal structure, maintains a rolling KV-cache with joint visual guidance, and distils denoising into a single step for unbounded-length inference. It satisfies A1 and A2, and we regard it as independent evidence that the design point characterised here is actively pursued rather than specific to our backbone. We are nonetheless unable to evaluate on it, either as a baseline or as a backbone: its official repository contains no implementation, and every item of its release checklist remains unchecked. Because DECACHE is training-free and operates on a backbone's internal cache and conditioning tensors, it cannot be applied without weights and code; no black-box or API access substitutes.



Figure 6: **Qualitative effect of Jacobi round count at the recovery chunk $t^\star$.** All columns share the identical rendering condition $x_{t^\star}$, isolating differences to what each method's memory carries forward. The vanilla baseline ($Y^{(0)}$, second column) still exhibits residual artifacts inherited from the corrupted interval preceding $t^\star$; it_1 (DECACHE) removes them. Iterating further (it_2, it_3, it_4) does not recover additional true detail: instead, increasingly sharp but spurious high-frequency texture accumulates.

We note the asymmetry between the two cases above: one falls outside our assumptions, the other satisfies them but is unavailable as an artefact. Since our method requires no training, adding a backbone-level evaluation on InfVSR is a matter of inference alone, and we commit to reporting it in the camera-ready should the model be released beforehand.