ENCOREMEM: RE-ENCOUNTER VERIFICATION OF REGION MEMORY FOR TRAINING-FREE LONG VIDEO GENERATION

Anonymous authors
Paper under double-blind review



Figure 1: **Long video generation with ENCOREMEM.** One frame per ten-second segment of one multi-prompt script of the Table 1 benchmark, with new requests in red; row labels summarize each rollout, red boxes mark color drift and inconsistent or duplicated faces, and green boxes mark faces kept consistent.

ABSTRACT

Autoregressive video models generate a long video one chunk at a time, conditioning each chunk on a cache of what has already been generated. Keeping the earliest frames in this cache helps limit degradation. These frames hold only what appeared at the start, so the rest of the cache must decide which later content to keep and for how long. Existing methods choose what to keep by its current or predicted relevance, or by learned representations. We instead keep stored content only if later frames, which also follow the current prompt, return to it. We introduce ENCOREMEM, which stores newly appearing parts of frames after they persist over several chunks and keeps each part provisionally. Whether a part returns is detected with the features of the generator, and when memory fills, the part that returned least often in recent chunks is replaced. The rule needs no training, additional model, retrieval at read time, or cache rebuild when the prompt changes. In minute-long video generation with both multi and single prompts, ENCOREMEM achieves competitive overall performance.

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.

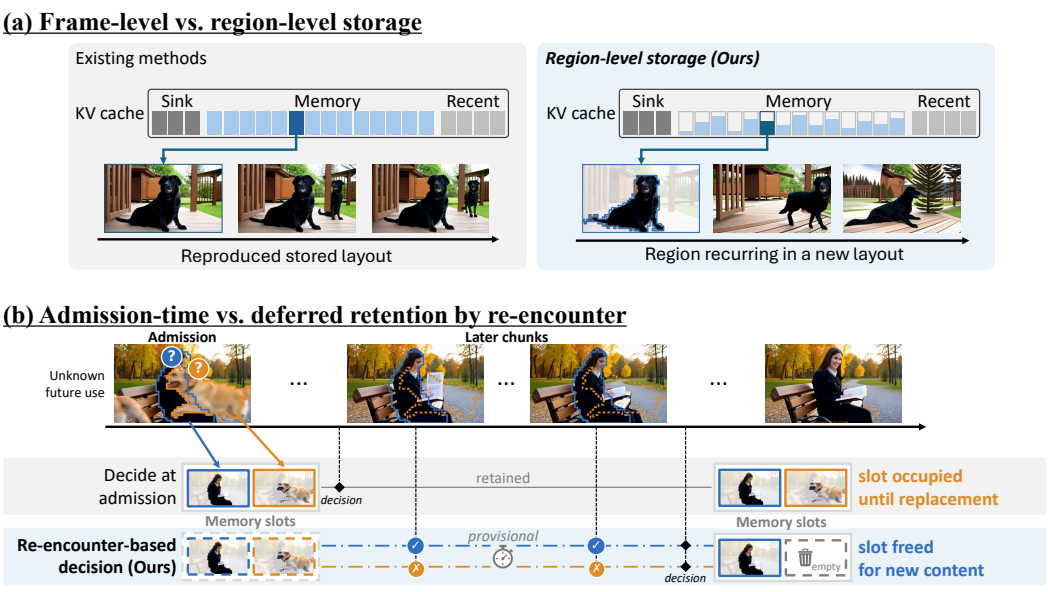

Figure 2: **Motivation of EncoreMem.** (a) A stored frame fixes its layout (left), whereas a stored region remains flexible to new layouts (right). (b) At admission, the future use of a region is unknown. A decision at admission retains both regions, so the region of the dog occupies a slot until replacement after the dog leaves the view (top), whereas EncoreMem keeps both regions provisional and releases the region of the dog, which later chunks do not re-encounter, freeing its slot for new content (bottom). Frames are from one prompt and seed.

1 Introduction

Recent advances in video generation have substantially improved the visual quality and motion of short video clips (Wan et al., 2025; Kong et al., 2024). However, many applications need videos that go beyond short clips, including educational and cinematic content and interactive storytelling (Yang et al., 2026). For these applications, a long video must preserve important visual content while allowing scenes and events to evolve, in addition to maintaining high visual quality (Yang et al., 2026; Wu et al., 2026).

A practical approach to long video generation is to produce the video incrementally, with each new frame or chunk conditioned on previously generated content (Chen et al., 2024; Yin et al., 2025; Huang et al., 2025). Recent transformer-based systems maintain representations of this past content in a key-value (KV) cache, which provides context for generating the next chunk (Yin et al., 2025; Huang et al., 2025). This process makes long video generation practical, but generated outputs also become part of the context for future generation, allowing small errors to accumulate over time. At the same time, the amount of past context cannot grow indefinitely as the video becomes longer. Recent content is important for local continuity, while earlier content can be necessary for maintaining subjects and scenes over longer periods. Long video generation therefore requires deciding what past information to store and how long to retain it under a limited context budget.

Existing ways of using this budget face two difficulties. Retaining early frames as a fixed reference can reduce long-term drift (Yi et al., 2026; Li et al., 2026b). However, repeatedly attending to stored frames also preserves their background, viewpoint, and spatial arrangement, which can restrict how the scene changes over time (Li et al., 2026a) (Figure 2a). The remaining memory capacity should therefore preserve useful content from the past without forcing later generation to reproduce an entire earlier frame. Existing methods address this problem by selecting past frames or tokens according to estimated relevance or importance (Meng et al., 2026; Yi et al., 2026; Luo et al., 2026; Zhao et al., 2026b). The second difficulty is that long-term usefulness is not always known when content first appears (Luo et al., 2026). A subject that later becomes important may initially appear no more relevant than transient scenery (Figure 2b). The decision to retain content therefore need not be final at the time of storage.

Building on this premise, we introduce ENCOREMEM, a training-free memory mechanism for long video generation. Instead of preserving an entire frame, ENCOREMEM identifies newly appearing visual content and stores only the corresponding region. The stored region is kept provisionally, and as generation proceeds, the method uses later re-encounters of similar visual content to decide whether the region should remain in memory. Existing methods can also revise what they keep in memory as generation proceeds (Meng et al., 2026; Dou et al., 2026), whereas ENCOREMEM bases the decision to keep a region only on whether later frames re-encounter it. All decisions use only representations from the frozen generator, so the method needs no training, auxiliary model, retrieval at read time, or cache reconstruction when the prompt changes.

Overall, this work makes two contributions for long video generation. The first is *region-level storage*, which stores the part of a frame around newly appearing content instead of the whole frame and delimits it with the features of the frozen generator rather than an external model. The second is *deferred retention*, which separates initial admission from long-term retention and updates the memory decision using evidence from later re-encounters. When a multi-prompt asks for an earlier scene again, storing regions lets the video follow the request more closely than storing whole frames or storing nothing and repeat earlier layouts less than storing whole frames, and deferred retention frees the memory held by content that does not return (Section 4.4). Without any training, ENCOREMEM performs comparably to methods trained for long video generation, and when the prompt changes, its consistency and aesthetic quality drop less over time than those of the other long-video methods (Table 1; Appendix E.1; Figure 1).

2 RELATED WORK

Autoregressive Long Video Generation. Autoregressive approaches extend short-clip video models (Wan et al., 2025; Kong et al., 2024) by generating each chunk conditioned on previously generated content (Chen et al., 2024; Yin et al., 2025; Huang et al., 2025), but errors can propagate to later chunks over long rollouts (Yin et al., 2025; Huang et al., 2025; Li et al., 2026b). Training-based approaches expose the model to generated context or adapt it for longer generation (Huang et al., 2025; Yang et al., 2026), whereas training-free approaches control which parts of the previous context remain available at inference time (Li et al., 2026b; Yi et al., 2026; Meng et al., 2026). Because only limited context can be kept, choosing which context to keep is central to both.

Memory for Long-Horizon Generation. As in KV-cache management for language models (Xiao et al., 2024; Zhang et al., 2023; Li et al., 2024), existing methods organize the cache by temporal role (Li et al., 2026b; Zhao et al., 2026b; Chen et al., 2026), compress older context into compact representations (Kim et al., 2026; Lu et al., 2026), or select what to keep from signals inside the generator such as attention and similarity (Yi et al., 2026; Meng et al., 2026; Zhao et al., 2026b), from entity or object-centered structure (Liu et al., 2026; Dou et al., 2026), or by retrieval at read time (Ji et al., 2025). Concurrent work stores newly appearing regions found with an optical-flow model at their first appearance (Xu et al., 2026), admits frames that leave the recent window to a memory bank by their novelty (Shi et al., 2026), evicts tokens whose neighborhoods become crowded by later blocks (Zhao et al., 2026a), or routes subject and scene queries to different parts of the history (Li et al., 2026a). Other methods revise what the memory holds by attention-based relevance and temporal diversity (Meng et al., 2026), use learned object slots with prompt-aware retrieval (Dou et al., 2026), or estimate which tokens future queries will attend to (Luo et al., 2026). Unlike these methods, ENCOREMEM delimits stored regions with the features of the frozen generator rather than an external model and keeps a region only if later frames re-encounter it. Appendix C discusses them further.

3 METHOD

3.1 OVERVIEW

We consider an autoregressive video generator that conditions each new chunk on a fixed-capacity key-value (KV) cache of previously generated content, and the generator remains frozen throughout. Following methods that separate early, retained, and recent context (Li et al., 2026b; Zhao et al., 2026b; Chen et al., 2026; Meng et al., 2026), we organize the cache into three parts, written as

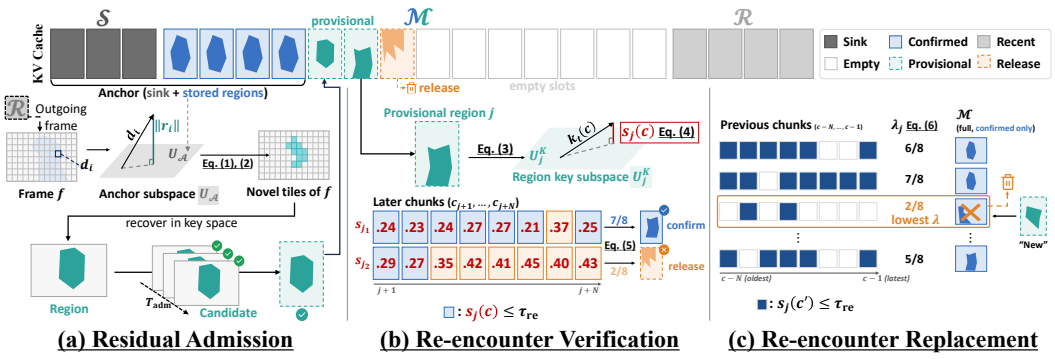

Figure 3: **Overview of EncoreMem.** The KV cache is partitioned into a fixed sink, region memory, and recent window. (a) Residual Admission identifies content unexplained by the sink and stored regions, recovers the corresponding region from pre-RoPE keys, and provisionally admits it after persistence over consecutive chunks. (b) Re-encounter Verification tests whether later chunks re-encounter each provisional region and confirms or releases it based on the accumulated evidence. (c) Re-encounter Replacement replaces the confirmed region with the lowest recent re-encounter rate when the memory is full and a new region is admitted.

$[\mathcal{S}\|\mathcal{M}\|\mathcal{R}]$: a sink $\mathcal{S}$ that keeps the first frames of the rollout as an attention sink (Xiao et al., 2024), a memory $\mathcal{M}$, and a recent window $\mathcal{R}$ of the latest frames. The sink and the recent window provide stable and local references, while EncoreMem operates only on $\mathcal{M}$ to preserve selected content from the longer history. The memory is divided into slots, each holding at most one stored region, the KV tokens of one region of a generated frame; a stored region is either *provisional* or *confirmed*. All operations run once per generated chunk on its middle latent frame, denoted f.

EncoreMem manages the memory with three components (Figure 3). *Residual Admission* (RA) decides what enters the memory, from the residual of new content against the sink and the stored regions. *Re-encounter Verification* (RV) decides what stays, from whether later chunks re-encounter the stored content. *Re-encounter Replacement* (RR) decides what leaves when the memory is full, from the same re-encounter evidence. All three components judge stored content only by whether it recurs in later frames, measured with tensors that the generator already computes.

3.2 Residual Admission

The memory should add content that the sink and the stored regions cannot already supply, and it should not spend a slot on content that is present only for a moment: every stored region is read by attention in the chunks that follow, so momentary content would keep influencing later frames after it has left the scene while occupying a slot that returning content could use. Residual Admission therefore looks for such content in each new frame and stores it as a region. It requires three properties before storing a region: it grows from content that the *anchors*, namely the sink together with every stored region, provisional or confirmed, do not explain; it is a coherent part of the frame rather than the whole frame; and it persists over consecutive chunks. The three steps below establish these properties in turn.

We first measure, for each part of f, how much of its content the anchors leave unexplained. We divide f into tiles, describe each tile by a unit-norm descriptor d_i computed from its latents, and represent the anchors by the SVD basis $U_{\mathcal{A}}$ of their tile descriptors, which is rebuilt whenever the stored regions change. The residual of a tile is the part of its descriptor outside this basis, and its novelty grows with the residual:

$$r_i = d_i - U_{\mathcal{A}} U_{\mathcal{A}}^{\top} d_i, \qquad \nu_i = 1 - \sqrt{1 - \|r_i\|^2}. \tag{1}$$

Measuring against a subspace, rather than against individual anchor tiles, treats a tile as familiar whenever some combination of the appearance already held explains it. Tiles whose novelty exceeds a threshold computed over the tiles of the same frame are marked as novel tiles:

$$\tau^{(\nu)} = \mathrm{median}(\nu) + 0.5\ \mathrm{MAD}(\nu). \tag{2}$$

The threshold is relative to the current frame because the scale of residuals changes with the scene and the content already stored, and it uses the median and the median absolute deviation because a few extreme tiles do not move them.

Novel tiles mark where new content appears, but they tend to cover only its most distinctive parts, so the region to store is recovered around them in the key space of the generator. Connected novel tiles are grouped into sets, and a set is discarded before recovery if it contains a single tile, or if it largely overlaps a stored region and would therefore bring back content already held. The pre-RoPE keys of an intermediate layer are projected to a low-dimensional space and split into two clusters, which tends to separate the content of the frame from its surroundings. For each remaining set, the region is the union of the connected components that belong to the cluster holding most of its tokens and that intersect the set, and sets that yield the same region are merged. The region therefore follows the boundary that the generator itself draws in the current frame. A stored region should be a part of the frame rather than the frame, so the number of tokens per region is limited, and a region that exceeds the limit keeps the tokens of its tiles with the largest residual.

Novelty in a single frame also marks content that is present only briefly, such as background passing through the view or a transient rendering error, so a region is stored only after it persists. Each region is followed across chunks as a candidate: regions of consecutive chunks are matched by the cosine similarity of their value centroids, restricted to pairs whose centers are close and assigned by mutual nearest neighbors. A candidate is admitted once it has been observed in T_{adm} consecutive chunks, and it is written into an empty slot as a provisional stored region; Section 3.4 covers the case in which no slot is empty. The candidate of a stored region remains in matching and absorbs the regions of later chunks that match it, so the same content is not admitted twice, while a candidate that has not been stored is dropped after it goes unobserved for several chunks.

3.3 Re-encounter Verification

Residual Admission establishes that a region is new and persists, but whether it will be needed later cannot be measured at admission; background that stays in view for several chunks passes the same test as a subject that will return. Re-encounter Verification therefore defers the decision: a newly stored region stays provisional and is judged by later chunks. It represents the stored region, tests each later chunk for a re-encounter, and decides at the end of a fixed window.

A stored region is represented in the same key space in which its boundary was drawn. For a stored region j with tokens $\mathcal{T}_j$, we fit a subspace once, at admission, to the pre-RoPE keys of these tokens,

$$U_j^K = \mathrm{SVD}_{0.95}\left(\{k_t : t \in \mathcal{T}_j\}\right), \tag{3}$$

where U_j^K keeps the singular vectors that explain 95% of the energy.

For a later chunk c, we measure the smallest normalized residual of its token keys against this subspace,

$$s_j(c) = \min_t \frac{\left\| k_t(c) - U_j^K {U_j^K}^\top k_t(c) \right\|}{\|k_t(c)\|}, \tag{4}$$

and count chunk c as a re-encounter, $e_j(c) = 1$, when $s_j(c) \le \tau_{\mathrm{re}}$. Taking the minimum over all tokens of the frame lets the test detect returning content at any position in the frame.

A single re-encounter can occur by chance, so the decision uses the whole verification window of N chunks after admission at chunk c_j: the stored region is confirmed when

$$\frac{1}{N} \sum_{c=c_j+1}^{c_j+N} e_j(c) \ge \tau_{\mathrm{conf}}, \tag{5}$$

and is released otherwise, which empties its slot. Here τ_{re} sets when a single chunk counts and τ_{conf} how often chunks must count; both are calibrated once on development rollouts and fixed for evaluation (Appendix A.4). Once the cache is full, a provisional stored region is read by attention in the same way as a confirmed one during its window, so that it is judged on the rollout in which it takes part (Appendix A.2); being read does not by itself make it recur, since each chunk also follows the prompt and the recent window (Figure 4b). Release keeps the candidate of the region together with its persistence count, so content that continues to appear is admitted again without rebuilding persistence and is judged over a new window.

3.4 Re-encounter Replacement and Readout

The memory has a fixed number of slots, so a new region may find no empty slot, and a confirmed stored region must then give way. Re-encounter Replacement makes this choice with the same evidence that confirmation used, which continues to accumulate after a stored region is confirmed. For each confirmed stored region we maintain its recent re-encounter rate over the N chunks before the current chunk c, since admission and replacement in chunk c precede its re-encounter test,

$$\lambda_j = \frac{1}{N} \sum_{c'=c-N}^{c-1} e_j(c'), \tag{6}$$

and the confirmed stored region with the smallest λ_j, the earliest admitted among ties, is replaced by the new region. Provisional stored regions are not replaced before their verification window closes, since they would then go unjudged; if no confirmed stored region exists, the admission is postponed.

Every stored region, provisional or confirmed, is part of the KV cache that generates later chunks, and attention reads all of them; EncoreMem performs no retrieval or selection at read time. Before the cache first fills, the memory positions hold previously generated frames as in an ordinary cache, and stored regions are read once the cache is full (Appendix A.2). Because no rule reads the text prompt, a change of prompt triggers no retrieval, reselection, or cache reconstruction, and the memory changes only through admission, confirmation or release, and replacement. The stored content and the evidence for keeping it both come from generated frames, so the memory could reinforce generation errors; to limit this, the same content is not admitted twice while stored, a stored region is read together with the sink, the recent window, and the prompt, and a provisional stored region is released when later frames do not re-encounter it within its window.

4 Experiments

4.1 Experimental Setup

We run all configurations (Appendix A) on a frozen autoregressive video generator distilled for streaming synthesis (Huang et al., 2025) from a public text-to-video backbone (Wan et al., 2025), and we run every compared method from its public implementation and weights (Appendix B). We evaluate 100 six-segment multi-prompts (Ji et al., 2025) and 128 MovieGen prompts for single-prompt generation (Polyak et al., 2024; Yi et al., 2026). We also build 30 scene-recall prompts that request earlier scenes again, taking their segments by a fixed rule from the first segments of the multi-prompts (Section 4.4; Appendix B.3).

4.2 Quantitative Results

We first evaluate rollouts whose prompt changes five times, so that each segment must follow a new prompt while the video stays visually stable (Table 1). Agreement with the current prompt declines over the rollout for every method, least for the two training-free methods and most for the base generator (last column). At a comparable decline, EncoreMem records higher Consistency and Aesthetic than Deep Forcing, and both paired differences over the 100 multi-prompts have 95% bootstrap intervals above zero. Measured the same way from the first to the last segment, consistency and aesthetic quality also decline least for EncoreMem among the compared methods other than the base generator (Appendix E.1). The base generator, trained on short clips, is shown for reference and not ranked; its highest Consistency comes with the lowest prompt agreement, since consistency compares frames within two-second windows and does not penalize a rollout that stops following new prompts, much as stagnating videos receive high temporal-quality scores (Yesiltepe et al., 2026). Tables 1 and 4 compare complete methods, whose caches also differ (Appendix B.1), and Section 4.4 isolates the memory within one cache.

4.3 Qualitative Results

The scores of Sections 4.2 and 4.5 summarize whole rollouts and show neither how content is carried along a rollout nor what the memory holds, so we inspect rollouts directly (more in Appendix E.2).

Table 1: **Quantitative comparison with multi-prompts.** Sixty-second rollouts on MemFlow multi-prompts, generated and scored by us (Appendix B); CLIP score measures each ten-second segment against its prompt, and Δ_{6-1} is the last minus the first. The base generator (–) is not ranked.

Method	Training-free	Consistency	Aesthetic	Quality	CLIP score 0-10s	10-20s	20-30s	30-40s	40-50s	50-60s	$\Delta_{6-1}\uparrow$
Self Forcing (Huang et al., 2025)	–	97.20	57.39	80.88	26.06	22.79	21.11	18.48	17.13	15.38	−10.68
MemFlow (Ji et al., 2025)	✗	**96.90**	59.05	81.34	**26.32**	24.20	23.26	23.23	23.84	22.99	−3.33
LongLive (Yang et al., 2026)	✗	96.81	60.53	81.81	26.21	24.89	24.29	**24.22**	**24.76**	23.95	−2.26
Deep Forcing (Yi et al., 2026)	✓	96.11	59.71	**83.21**	25.99	24.94	24.27	23.97	24.13	24.17	−1.82
EncoreMem (Ours)	✓	96.81	**60.95**	83.01	26.02	**25.67**	**24.59**	23.41	24.23	**24.22**	**−1.80**

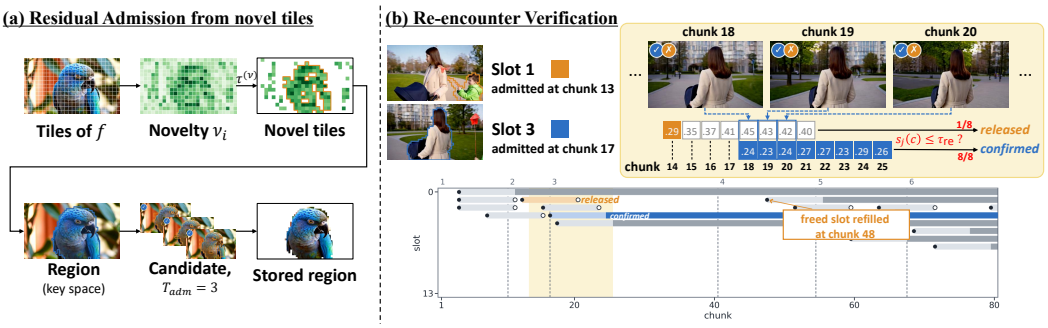

Figure 4: **Memory visualization of EncoreMem.** (a) Residual Admission: tile novelty ν_i, novel tiles above $\tau^{(\nu)}$, the region recovered in key space, and the region stored after it persists over $T_{\text{adm}} = 3$ chunks. (b) Re-encounter Verification: scores $s_j(c)$ of two stored regions over the $N = 8$ chunks after admission, with red highlights marking $s_j(c) \le \tau_{\text{re}}$, so slot 3 is confirmed and slot 1 released; below, slot states over the rollout (light: provisional, dark: confirmed, filled and open circles: admissions and releases, dashed lines: prompt switches, shaded band: the windows above).

With multi-prompts, the methods differ in whether a newly requested subject appears on time and stays afterward, as Figure 1 shows on a multi-prompt whose second segment introduces a research partner who remains until the end. The base generator never introduces the partner and drifts into saturated colors from the third segment on. MemFlow keeps the scientist in nearly the same pose throughout and shows the partner only at the edge of the frame. LongLive and Deep Forcing introduce the partner, but LongLive changes the face of the scientist across segments and Deep Forcing renders an overlapping face; none of the three renders the high-five requested in the last segment. EncoreMem introduces the partner in the second segment, follows the later requests from looking at the tablet to raising a hand for the high-five, and keeps both people as the same subjects throughout.

With a single prompt, the difficulty shifts from following new requests to keeping one coherent subject over the minute (Figure 5). The base generator degrades into saturated artifacts in the second half. MemFlow and LongLive keep the scene stable but render a second copy of the runner in the fifth segment, and Deep Forcing, which moves most, shifts the sky color so strongly in the same segment that the scene washes out. EncoreMem keeps a single runner with the same appearance throughout while the viewpoint changes and the rocket plume appears as requested.

To check that the modules behave as designed on actual rollouts, we trace their decisions in Figure 4. Residual Admission stores the region of the bird, which the anchors do not explain and which persists over T_{adm} chunks, rather than the whole frame (Figure 4a). Re-encounter Verification confirms the region of a woman that later chunks keep re-encountering and releases the region of a boy who leaves the view, whose slot is later reused (Figure 4b).

4.4 Scene Recall and Ablation

The standard settings do not request an earlier scene again after intervening prompts, which is where a memory has to keep earlier content while the video continues to change. We therefore evaluate 30 scene-recall prompts of the form A–B–C–A'–B'–C', in which the last three segments repeat



Figure 5: **Qualitative comparison with a single prompt.** One frame per ten-second segment of a sixty-second rollout for the prompt below; red boxes mark saturated artifacts, a duplicated subject, and a color shift, and all clips are from the evaluation of Table 4.

Table 2: **Ablation on scene recall.** Thirty scene-recall prompts per configuration (Appendix B.3); RA, RV, and RR mark Residual Admission, Re-encounter Verification, and Re-encounter Replacement (Section 3.1). Crop similarity measures the appearance of recalled objects between a segment and its repeat, flow and Dynamic the motion of the repeated segments, ViCLIP their agreement with the repeated prompt, and layout repetition how closely frames reproduce the arrangement of earlier frames.

Configuration	Unit	RA	RV	RR	Crop sim. ↑	Flow ↑	Dynamic ↑	ViCLIP ↑	Layout rep. ↓
Full (EncoreMem)	region	✓	✓	✓	0.615	**4.38**	**28.33**	**10.23**	**0.688**
w/o RV	region	✓	✗	✓	0.632	4.32	26.33	10.00	**0.688**
w/o memory	–	✗	✗	✗	0.671	1.96	24.00	8.99	0.708
Frame memory	frame	✗	✗	✗	**0.725**	2.34	27.67	6.41	0.812

the first three prompts, and compare B' and C' with their first occurrences because A can also be represented by the frozen sink (Appendix B.3).

To test the two decisions of Section 1, what to store and how long to retain it, we remove the components of EncoreMem in turn (Table 2). The configuration w/o RV confirms every provisional stored region, w/o memory leaves the memory unread, and frame memory stores whole frames without RA and RV, so it compares memory designs rather than isolating the storage unit.

On what to store, regions let the rollout answer a repeated prompt with a new layout: frame memory reproduces earlier layouts most and follows the repeated prompt least, and its highest crop similarity comes with lower motion, which raises crop similarity as it raises consistency (Section 4.2; Figure 2a). Compared with w/o memory, the region memory moves more and repeats earlier layouts less, and once the dependence of crop similarity on motion is removed, its appearance similarity is not measurably different from that of the other configurations (Appendix B.3).

For how long to retain it, the comparison with w/o RV tests the argument of Section 1 that a decision at admission keeps content that does not return (Figure 2b). With verification, regions that are not re-encountered are released and confirmed regions are replaced less often, whereas confirming every region leaves such regions in memory until capacity forces a replacement (Table 3; Figure 4b).

Table 3: **Memory events.** Counts over the 30 scene-recall clips of each configuration; releases follow a failed verification, replacements remove the confirmed stored region with the lowest recent re-encounter rate, and unjudged regions have no decision by the end of the rollout.

Configuration	Admitted	Confirmed	Released	Unjudged	Replaced
Full (EncoreMem)	559	447	64	48	111
w/o RV	577	515	0	62	179

Table 4: **Quantitative comparison with a single prompt.** Sixty-second rollouts on 128 MovieGen prompts scored with VBench-Long, generated and scored by us (Appendix B.1); marks follow Table 1. EncoreMem runs on the Self Forcing generator of the first row.

Method	Training-free	Dynamic	Imaging	Motion	Background	Aesthetic	Subject	Overall Cons.
Self Forcing (Huang et al., 2025)	–	28.54	67.37	98.23	96.64	56.37	97.07	21.14
MemFlow (Ji et al., 2025)	✗	37.58	69.67	**98.79**	**96.58**	60.88	**97.42**	24.34
LongLive (Yang et al., 2026)	✗	40.39	68.98	98.75	96.52	**61.63**	97.35	**24.63**
Deep Forcing (Yi et al., 2026)	✓	**65.00**	68.46	97.73	95.57	59.76	95.97	23.64
EncoreMem (Ours)	✓	48.54	**70.39**	98.30	96.28	60.84	97.04	23.82

Because replacement in w/o RV still uses the re-encounter rate, the two differ mainly in how long regions that are not re-encountered stay, and the measurements of Table 2 change little.

4.5 Single-Prompt Generation

The memory is designed for rollouts whose content changes, so we also check that it does not degrade generation when the prompt stays fixed (Table 4). With a single prompt, the methods trade motion against stability: the training-based methods move least with the highest smoothness and consistency, whereas Deep Forcing moves most with the lowest smoothness and consistency and, among the training-free methods, the largest decline along the rollout (Appendix E.1; Figure 5). Dynamic counts the two-second subclips whose optical flow exceeds a threshold (Huang et al., 2024; 2026) and does not separate coherent motion from unstable change, so it is read together with motion smoothness. Without training, EncoreMem stays close to the training-based methods in smoothness and consistency while recording more motion and the highest Imaging score, and it exceeds Deep Forcing on every dimension except Dynamic (Table 4).

Because EncoreMem moves less than Deep Forcing, one may ask whether it simply suppresses motion. It moves more than its base generator while raising Imaging and aesthetic quality, and on scene recall the region memory moves more than the same cache without memory (Table 2); its gap to Deep Forcing lies where Deep Forcing loses smoothness and consistency.

5 Conclusion

EncoreMem manages the long-term memory of a streaming KV cache with two choices that use only tensors of the frozen generator. Region-level storage writes the tokens of a region grown from content that the sink and the stored regions leave unexplained, so a stored region keeps what appeared without the arrangement of its frame. Deferred retention admits each region provisionally and keeps it only if later chunks re-encounter it in pre-RoPE key space, and what stays in memory is thus decided by what the rollout goes on to produce rather than by an estimate made when the content first appears. Neither choice requires training, an auxiliary model, retrieval at read time, or a cache rebuild when the prompt changes; the resulting method performs comparably to methods trained for long generation, and on repeated scene requests its memory reproduces earlier layouts less than a memory of whole frames while following the repeated prompt more closely. More broadly, these results suggest that memory for streaming video generation can be managed by observing whether content returns, which may carry over to other generators that expose their KV cache.

AI USE STATEMENT

In this work, we used generative AI tools to help propose and refine hypotheses, to propose design options and give feedback on the method and the experiments, to implement the method and the evaluation, to help interpret results, to assemble the scene-recall prompts by recombining prompts of existing benchmarks, to write the prompts of the qualitative examples in Figure C of Appendix E.2, and to translate manuscript text from notes of the authors. Theoretical models, mathematical claims, proofs, and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for readability, to search related literature, to format references, and to prepare figure candidates. All research decisions, including the final design of the method and the experiments, were made by the authors, who reviewed the AI-assisted work used in this paper. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not collect new human-subject data. Our experiments use publicly available benchmark prompts and pretrained generative models. Our method is training-free and does not introduce additional training data or newly trained model weights.

REPRODUCIBILITY STATEMENT

Implementation details and hyperparameters are given in Appendix A, and benchmarks, generation settings, and metrics in Appendix B. Every method in the reported tables, including the baselines, is run by us from its public code and checkpoint on the same prompts and seeds and scored with the same metrics (Appendix B).

REFERENCES

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets Full-Sequence diffusion. In *NeurIPS*, 2024.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, Weiming Xiong, Wei Wang, Nuo Pang, Kang Kang, Zhiheng Xu, Yuzhe Jin, Yupeng Liang, Yubing Song, Peng Zhao, Boyuan Xu, Di Qiu, Debang Li, Zhengcong Fei, Yang Li, and Yahui Zhou. SkyReels-V2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

Shuo Chen, Cong Wei, Sun Sun, Tiancheng Shen, Ping Nie, Kai Zou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context forcing: Consistent autoregressive video generation with long context. In *ICML*, 2026.

Weijia Dou, Hui Li, Jiahao Cui, Lei Zhou, Jingdong Wang, and Siyu Zhu. SlotMemory: Object-Centric KV memory for streaming Long-Video generation. *arXiv preprint arXiv:2605.31033*, 2026.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *ICLR*, 2024.

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.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the Train-Test gap in autoregressive video diffusion. In *NeurIPS*, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua

Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(3):3268–3285, 2026. doi: 10.1109/TPAMI.2025.3633890.

Sihui Ji, Xi Chen, Shuai Yang, Xin Tao, Pengfei Wan, and Hengshuang Zhao. MemFlow: Flowing adaptive memory for consistent and efficient long video narratives. *arXiv preprint arXiv:2512.14699*, 2025.

Youngrae Kim, Qixin Hu, C.-C. Jay Kuo, and Peter A. Beerel. MemRoPE: Training-Free infinite video generation via evolving memory tokens. In *ECCV*, 2026.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Chen Li, Peng Zhang, Hanyu Zhou, Jialong Zuo, Fei Wang, Daiguo Zhou, Nong Sang, and Changxin Gao. Tether the subject, release the scene: Query-aware memory routing for long-horizon autoregressive video generation. *arXiv preprint arXiv:2608.26902*, 2026a.

Haodong Li, Shaoteng Liu, Zhe Lin, and Manmohan Chandraker. Rolling sink: Bridging Limited-Horizon training and Open-Ended testing in autoregressive video diffusion. *arXiv preprint arXiv:2602.07775*, 2026b.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *NeurIPS*, 2024.

Jinzhuo Liu, Jiangning Zhang, Wencan Jiang, Yabiao Wang, Dingkang Liang, Zhucun Xue, Ran Yi, and Yong Liu. Advancing narrative long video generation via Training-Free Identity-Aware memory. *arXiv preprint arXiv:2605.18733*, 2026.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *NeurIPS*, 2023.

Yu Lu, Yuanzhi Liang, Linchao Zhu, and Yi Yang. FreeLong: Training-Free long video generation with SpectralBlend temporal attention. In *NeurIPS*, 2024.

Yu Lu, Junjie Yang, Piotr Koniusz, YuXin Song, and Yi Yang. FadeMem: Distance-Aware memory consolidation for autoregressive video diffusion. *arXiv preprint arXiv:2606.10671*, 2026.

Jiayi Luo, Qiyan Liu, Tengyang Wang, JunHao Liu, Jiayu Chen, Cong Wang, Hanxin Zhu, Chen Gao, Xiaobin Hu, Qingyun Sun, and Zhibo Chen. Future forcing: Future-aware training-free KV cache policy for autoregressive video generation. *arXiv preprint arXiv:2605.30083*, 2026.

Yu Meng, Xiangyang Luo, Letian Li, Wenyuan Jiang, Chen Gao, Xinlei Chen, Yong Li, and Xiao-Ping Zhang. TetherCache: Stabilizing autoregressive Long-Form video generation with gated recall and trusted alignment. *arXiv preprint arXiv:2606.13035*, 2026.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, Dingkang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu,

Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Artsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie Gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Haonan Qiu, Menghan Xia, Yong Zhang, Yingqing He, Xintao Wang, Ying Shan, and Ziwei Liu. FreeNoise: Tuning-Free longer video diffusion via noise rescheduling. In *ICLR*, 2024.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

Sand.ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, Weifeng Luo, Xiaoyang Kang, Yuchen Sun, Yue Cao, Yunpeng Huang, Yutong Lin, Yuxin Fang, Zewei Tao, Zheng Zhang, Zhongshu Wang, Zixun Liu, Dai Shi, Guoli Su, Hanwen Sun, Hong Pan, Jie Wang, Jiexin Sheng, Min Cui, Min Hu, Ming Yan, Shucheng Yin, Siran Zhang, Tingting Liu, Xianping Yin, Xiaoyu Yang, Xin Song, Xuan Hu, Yankai Zhang, and Yuqiao Li. MAGI-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Shuwei Shi, Zhen Li, Muyao Niu, Chuanhao Li, Bo Zheng, Kaipeng Zhang, and Yinqiang Zheng. Surprise forcing: What to remember, when to skip in long video generation. In *ECCV*, 2026.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-Aware sparsity for efficient Long-Context LLM inference. In *ICML*, 2024.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual descriptions. In *ICLR*, 2023.

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

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. InternVid: A large-scale video-text dataset for multimodal understanding and generation. In *ICLR*, 2024.

Mingqiang Wu, Weilun Feng, Zhefeng Zhang, Haotong Qin, Yuqi Li, Guoxin Fan, Xiaokun Liu, Zhulin An, Libo Huang, Yongjun Xu, and Chuanguang Yang. Echo-Forcing: A scene memory framework for interactive long video generation. *arXiv preprint arXiv:2605.16003*, 2026.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

Haiyang Xu, Zheng Ding, and Zhuowen Tu. RECAP-Forcing: Retaining content appearances for long video generation. *arXiv preprint arXiv:2608.26671*, 2026.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time interactive long video generation. In *ICLR*, 2026.

Hidir Yesiltepe, Tuna Han Salih Meral, Adil Kaan Akan, Kaan Oktay, and Pinar Yanardag. Infinity-RoPE: Action-Controllable infinite video generation emerges from autoregressive Self-Rollout. In *CVPR*, 2026.

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep forcing: Training-Free long video generation with deep sink and participative compression. In *ICML*, 2026.

Shengming Yin, Chenfei Wu, Huan Yang, Jianfeng Wang, Xiaodong Wang, Minheng Ni, Zhengyuan Yang, Linjie Li, Shuguang Liu, Fan Yang, Jianlong Fu, Ming Gong, Lijuan Wang, Zicheng Liu, Houqiang Li, and Nan Duan. NUWA-XL: Diffusion over diffusion for eXtremely long video generation. In *ACL*, 2023.

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 *CVPR*, 2025.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in Next-Frame-Prediction video diffusion models. In *NeurIPS*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023.

Min Zhao, Guande He, Yixiao Chen, Hongzhou Zhu, Chongxuan Li, and Jun Zhu. RIFLEx: A free lunch for length extrapolation in video diffusion transformers. In *ICML*, 2025.

Wenqu Zhao, Xuemin Chi, Xin Zhang, Guoqing Ma, Baorun Li, Jianjie Fang, Peizhi Tang, Chen Gao, and Wei Wu. DensityKV: Density-guided KV cache compression for long video generation. *arXiv preprint arXiv:2608.27922*, 2026a.

Zengqun Zhao, Yanzuo Lu, Ziquan Liu, Jifei Song, Jiankang Deng, and Ioannis Patras. Relax forcing: Relaxed KV-Memory for consistent long video generation. In *BMVC*, 2026b.

ENCOREMEM: RE-ENCOUNTER VERIFICATION OF REGION MEMORY FOR TRAINING-FREE LONG VIDEO GENERATION

Supplementary Material

TABLE OF CONTENTS

A. Implementation Details . . . 1
A.1 Complete Procedure . . . 1
A.2 Cache Write and State Transitions . . . 1
A.3 Hyperparameters . . . 2
A.4 Threshold Calibration . . . 2
B. Evaluation Details . . . 2
B.1 Benchmarks and Generation Settings . . . 3
B.2 Metrics . . . 4
B.3 Scene-recall Benchmark . . . 4
C. Extended Related Work . . . 5
D. Limitations . . . 5
E. Additional Results . . . 6
E.1 Change Along the Rollout . . . 6
E.2 Additional Qualitative Results . . . 6

A IMPLEMENTATION DETAILS

The main text defines Residual Admission, Re-encounter Verification, and Re-encounter Replacement at the level needed to describe the method. This section gives the execution order, the state of a stored region, and the values of the hyperparameters.

A.1 COMPLETE PROCEDURE

Algorithm A gives the memory update around one generated chunk, with admission and replacement preceding verification; the generator remains frozen, and only the content and state of the KV cache change. Besides the quantities of Section 3, it writes n for the number of slots, $D(\cdot)$ and $K_{(\cdot)}$ for the tile descriptors and pre-RoPE keys of a token set, $\mathrm{CC}(\cdot)$ for connected components on the token grid, IoU for bounding-box overlap, $\ell_t \in \{1, 2\}$ for the key-space cluster of token t, Match for the mutual-nearest assignment of regions to candidates by value-centroid cosine within the center-distance gate, top_L for keeping the L tokens whose tiles have the largest residual, and $\mathcal{K}_{\mathcal{J}}$ for the candidates of stored regions; the minimum over (λ_j, c_j) is lexicographic.

A.2 CACHE WRITE AND STATE TRANSITIONS

A candidate records correspondence between regions observed across chunks, and a stored region holds, in one slot, the K/V tensors of the tokens of one admitted candidate. Admission places the stored region in the *provisional* state, and its K/V tensors are captured during the following generation step, before the corresponding attention layer reads the cache. The key basis U_j^K is fitted once from the captured pre-RoPE keys and stays fixed while the region is stored, whereas the anchor basis $U_{\mathcal{A}}$ is rebuilt whenever a region is admitted, released, or replaced. Within a chunk, admission and replacement run before verification, so the recent re-encounter rate used for replacement in chunk c covers chunks $c - N$ to $c - 1$.

Provisional and confirmed stored regions are read by attention in the same way; the state only determines how retention is decided. A provisional stored region that satisfies Eq. (5) at the end of

Algorithm A Memory update of ENCOREMEM around chunk c; equation numbers refer to Section 3.

Require: $\mathcal{J} = \{(\mathcal{T}_j, U_j^K, \sigma_j, c_j)\}$ with $\sigma_j \in \{\text{PROV}, \text{CONF}\}$; $\mathcal{K} = \{(\kappa, a_\kappa, m_\kappa)\}$

$\mathcal{M} \leftarrow \mathcal{M} \cup \{\text{KV}(\mathcal{T}_j) : c_j = c-1\}$; generate chunk c from $[\mathcal{S}\|\mathcal{M}\|\mathcal{R}]$; $f \leftarrow$ its middle latent frame

Residual Admission

$U_{\mathcal{A}} \leftarrow \text{SVD}_{0.95}\big(D(\mathcal{S}) \cup D(\bigcup_{j\in\mathcal{J}} \mathcal{T}_j)\big)$; $\mathcal{P} \leftarrow \{i : \nu_i > \tau^{(\nu)}\}$ ▷ Eqs. (1) and (2)

$\mathcal{G} \leftarrow \{g \in \text{CC}(\mathcal{P}) : |g| \geq 2,\ \max_j \text{IoU}(g, \mathcal{T}_j) \leq 0.5\}$

$R_g \leftarrow \bigcup\{C \in \text{CC}(\{t : \ell_t = \ell_g\}) : C \cap g \neq \emptyset\}$, $\ell_g = \text{mode}_{t\in g}\, \ell_t$, $g \in \mathcal{G}$

$\Pi \leftarrow \text{Match}(\{R_g\}, \mathcal{K})$; $(a_\kappa, m_\kappa) \leftarrow (a_\kappa{+}1,\ 0)$ if $\kappa \in \Pi$, else $(0,\ m_\kappa{+}1)$; add $(\kappa, 1, 0)$ for each unmatched R_g

$\mathcal{K} \leftarrow \mathcal{K} \setminus \{\kappa \notin \mathcal{K}_{\mathcal{J}} :\ m_\kappa \geq 3\}$

for $\kappa \in \mathcal{K} \setminus \mathcal{K}_{\mathcal{J}}$ with $m_\kappa = 0$ and $a_\kappa \geq T_{\text{adm}}$ **do**

  if $|\mathcal{J}| = n$ **then**

    $j^\star \leftarrow \arg\min_{j:\,\sigma_j=\text{CONF}} (\lambda_j, c_j)$; $\mathcal{J} \leftarrow \mathcal{J} \setminus \{j^\star\}$ ▷ Re-encounter Replacement, Eq. (6); skip κ if no $j^\star$

  end if

  $\mathcal{T} \leftarrow \text{top}_L(R_\kappa)$; $\mathcal{J} \leftarrow \mathcal{J} \cup \{(\mathcal{T},\ \text{SVD}_{0.95}(K_{\mathcal{T}}),\ \text{PROV},\ c)\}$ ▷ Eq. (3)

end for

Re-encounter Verification

for $j \in \mathcal{J}$ with $c_j < c$ **do**

  $e_j(c) \leftarrow \mathbf{1}\big[s_j(c) \leq \tau_{\text{re}}\big]$ ▷ Eq. (4)

  if $\sigma_j = \text{PROV} \wedge c = c_j + N$: $\sigma_j \leftarrow \text{CONF}$ if Eq. (5) holds, else $\mathcal{J} \leftarrow \mathcal{J} \setminus \{j\}$

end for

its window becomes confirmed, and one that fails is released and its slot becomes empty. Release keeps the candidate and its persistence count, so a released region that continues to be observed is admitted again without rebuilding persistence; any chunk in which a candidate is not observed resets its count. Replacement occurs only when an admission finds no empty slot and considers confirmed stored regions only; if every stored region is provisional, the admission is postponed. Before the cache first fills, the memory positions hold previously generated frames as in an ordinary cache, and stored regions are first read in the seventh chunk; for a region admitted earlier, part of its verification window precedes its first read.

A.3 HYPERPARAMETERS

Table A lists the values not given in the main text; all are fixed across every experiment.

A.4 THRESHOLD CALIBRATION

The thresholds τ_{re} and τ_{conf} are set once on nine development rollouts of sixty seconds, generated with verification and replacement disabled, and remain fixed for all experiments. In five of these rollouts, stored regions are labeled as object or background by fixed boxes marking the subject where one exists; the label is used only for calibration and is not available to the method. τ_{re} is the midpoint between the mean per-chunk projection ratios of the two classes, expressed as a residual share, and τ_{conf} is the midpoint between their mean re-encounter rates over the verification window at that τ_{re}. Three development prompts also appear among the 128 prompts of the single-prompt benchmark, with different seeds; none appears in the multi-prompts or scene-recall prompts.

B EVALUATION DETAILS

Every method, including the baselines, is generated and scored by us under the settings below, with the official code and checkpoint of each baseline. Values can therefore differ from those reported in the original papers.

Table A: **Hyperparameters of EncoreMem.** Values not given in the main text; token positions are measured on the 30×52 token grid of a latent frame.

Quantity	Value
Cache partition $\mathcal{S}/\mathcal{M}/\mathcal{R}$	3 / 14 / 4 latent frames
Chunk	3 latent frames; every operation runs once per chunk
Number of slots n	14, one stored region per slot
Rotary positions of $\mathcal{S}$ and $\mathcal{M}$	fixed positions directly before the recent window, re-rotated at every chunk
Tile	2×2 tokens (4×4 latent cell), 15×26 tiles per frame
Tile descriptor	clean latent of the cell, each channel standardized over the cell, L_2-normalized, 256 dimensions
Anchor basis $U_{\mathcal{A}}$	SVD without mean subtraction, 95% of the energy
Minimum set of connected novel tiles	2 tiles
Overlap with a stored region for discarding a set	bounding-box IoU > 0.5
Pre-RoPE keys for region recovery and re-encounter	layer 15 (0-based), keys of f from the pass that writes the cache
Key-space clustering	PCA to 3 dimensions, k-means with $k = 2$
Candidate matching	mutual highest value-centroid cosine within the distance gate
Center-distance gate	$\|p - p'\|_2 \leq (\delta + \delta')/2$, δ: bounding-box diagonal
Admission persistence T_{adm}	3 consecutive chunks
Dropping an unstored candidate	after 3 unobserved chunks
Token limit per stored region L	1040 tokens ($2/3$ of a frame)
Verification window N	8 chunks
Chunk-level threshold τ_{re}	0.3039, i.e. a projection ratio of at least 0.9527
Window-level threshold τ_{conf}	0.6653 (at least 6 of 8 chunks)

Table B: **Cache settings.** Cache attended by each method in steady state, in latent frames including the current chunk; the two values of EncoreMem are averages over the single-prompt and multi-prompt rollouts, and LongLive and MemFlow also recompute the recent cache at each prompt switch.

Method	Sink	Memory	Recent	Total
Self Forcing	0	–	21	21
MemFlow	3	memory bank, ≤ 3	9	15
LongLive	3	–	9	12
Deep Forcing	10	compressed, ≈ 2	4 or 7	16 or 19
EncoreMem	3	stored regions, 2.6 / 3.1	4	9.6 / 10.1

B.1 Benchmarks and Generation Settings

Multi-prompt benchmark. We use the 100 six-segment multi-prompts released with MemFlow (Ji et al., 2025), in which each segment gives a ten-second prompt, and generate one sixty-second video per prompt with seed 1 for every method. For every method the prompt switches at the first chunk boundary at or after each ten-second mark, and LongLive and MemFlow handle the switch as in their released implementations, and the other methods change only the text condition.

Single-prompt benchmark. We use 128 MovieGen prompts (Polyak et al., 2024) following Yi et al. (2026) and generate one sixty-second video per prompt with seed 1356145 for every method.

Common settings. All videos have 957 frames at 16 fps and 832×480 resolution, and each method uses the sampler of its released implementation.

Cache settings. Table B lists the cache that each method attends to. The cache of EncoreMem has room for 21 latent frames, but its memory holds only the tokens of stored regions.

Table C: **Example scene-recall prompt.** Scene-recall prompt 0 takes the first segments of multi-prompts 0, 1, and 2 as A, B, and C, and its last three segments repeat them in order.

Segments	Prompt
A, A' (0–10 s, 30–40 s)	A young wizard with a pointed hat and a sparkling robe is practicing magic in an ancient, stone-walled chamber. He waves a wooden wand, causing small objects on a table to float in the air. The chamber is lit by flickering torchlight, casting long shadows.
B, B' (10–20 s, 40–50 s)	A lone astronaut in a bulky white spacesuit is collecting rock samples on the desolate, red-sanded surface of Mars. The landscape is vast and empty, with distant craters under a thin, pinkish sky. A futuristic habitat dome is visible in the background.
C, C' (20–30 s, 50–60 s)	A female chef with a white toque and apron is meticulously plating a dish in a busy, stainless-steel restaurant kitchen. She carefully places a microgreen with tweezers. The kitchen is filled with the sounds of chopping and sizzling.

B.2 Metrics

For Table 1, Consistency is the mean of the VBench-Long (Huang et al., 2026) subject and background consistency dimensions, Aesthetic is its aesthetic quality dimension, and Quality aggregates seven VBench-Long dimensions with the normalization and weights of the official VBench quality score; each dimension is computed within two-second subclips and averaged over the video. The segment CLIP score is the video-text cosine similarity of ViCLIP (Wang et al., 2024), multiplied by 100, between eight uniformly spaced frames of each ten-second segment and the prompt of that segment, and Δ_{6-1} is the score of the sixth segment minus that of the first, averaged over multi-prompts. Table 4 reports seven VBench-Long dimensions, with Overall Consistency computed from the prompt text.

B.3 Scene-recall Benchmark

The scene-recall prompts request an earlier scene again after intervening segments, with six ten-second segments ordered as

$$A \to B \to C \to A' \to B' \to C',$$

where the last three segments repeat the first three prompts. We use 30 sixty-second scene-recall prompts, the same for every configuration, and evaluate B/B' and C/C' because the initial scene can also be represented by the sink. Scene-recall prompt i, counting from zero, takes as A, B, and C the first segments of multi-prompts $3i$, $3i{+}1$, and $3i{+}2$ in file order and repeats them as A', B', and C', and every configuration uses seed 1. Only first segments are used because later segments of a multi-prompt refer to earlier ones and do not stand alone. Table C shows the first scene-recall prompt.

Computation. Crop similarity uses one frame per segment, the fifth of eight frames sampled uniformly within the segment as bounded by its prompt switches. For each COCO class detected in both a segment and its repeat by Faster R-CNN (Ren et al., 2015) with score at least 0.7 and box sides of at least 16 pixels, the highest-scoring box is taken from each frame, both crops are resized to 224×224, and the cosine similarity of their DINOv2 ViT-B/14 (Oquab et al., 2024) class tokens is computed; the class scores are averaged, the B/B' and C/C' values are averaged, and a clip contributes only when both pairs yield a value. Crop similarity decreases as flow increases across clips ($r = -0.34$), so the two are read together. After removing the pooled linear trend of crop similarity on flow, the paired differences of Full are $+0.002$ ($[-0.044, +0.047]$) against w/o memory, -0.052 ($[-0.114, +0.009]$) against frame memory, and -0.009 ($[-0.051, +0.034]$) against w/o RV. ViCLIP and Dynamic are the segment CLIP score of Appendix B.2 and the VBench-Long dynamic degree over the fixed ten-second intervals of B' and C', averaged over the two. Flow applies RAFT (Teed & Deng, 2020) to the seven adjacent pairs of these eight frames of B' and of C', takes the median magnitude over pixels for each pair and the median over the pairs, and averages B' and C'. Layout repetition compares the DINOv2 patch tokens of each chunk-middle frame with those of every chunk-middle frame at least seven chunks earlier, averages the cosine over patch positions, keeps the maximum over the earlier frames, and averages it over the rollout; it is computed from generated frames only. Each column of Table 2 is the mean of the clip values over the scene-recall prompts.

Configurations. All configurations share the generator, the cache partition, the prompts, and the seed of Full, and differ only in how the memory positions are filled and read. The configuration w/o memory removes the memory positions from the keys and values once the cache first fills, so later chunks attend to the sink and the recent window alone. The configuration w/o RV runs Residual Admission and Re-encounter Replacement as in Full but confirms every provisional stored region when its verification window closes. Frame memory writes the KV tokens of every seventeenth latent frame that leaves the recent window into the memory positions in temporal order, which spreads the fourteen positions over a sixty-second rollout, without Residual Admission or Re-encounter Verification; a sixty-second rollout stores thirteen such frames, so the memory never fills, and positions not yet written keep the frames they held when the cache first filled.

C Extended Related Work

This section extends Section 2 to long video generation more broadly and to KV-cache management in language models.

Long video generation. Long videos have also been generated by tokenizing video for variable-length generation (Villegas et al., 2023), by coarse-to-fine diffusion over keyframes (Yin et al., 2023), or by extending short-clip diffusion models without training through noise rescheduling, spectral blending, or position-frequency adjustment (Qiu et al., 2024; Lu et al., 2024; Zhao et al., 2025). Diffusion-forcing and autoregressive models extend generation chunk by chunk, with memory modules (Henschel et al., 2025), compressed frame context (Zhang et al., 2025), diffusion forcing with multi-stage training for infinite-length generation (Chen et al., 2025), or large-scale chunk-wise training (Sand.ai et al., 2025).

Memory management. One line of work organizes the cache according to temporal roles, combining stable early context, recent context, and retained historical information (Li et al., 2026b; Zhao et al., 2026b; Chen et al., 2026). This separation allows different parts of the cache to support long-term stability and short-term continuity without retaining the full generation history. Another line reduces the cost of older context by summarizing or consolidating past information into compact memory representations (Kim et al., 2026; Lu et al., 2026). Such representations allow information from a longer temporal range to remain available under a fixed memory budget, although some details may be compressed or discarded. These approaches mainly determine how historical context is structured or compressed within the available capacity, not which specific content should be preserved.

Memory selection. One direction estimates the value of past information from signals already available inside the generator, such as the attention it receives from recent frames (Yi et al., 2026; Meng et al., 2026) or its similarity to retained context (Zhao et al., 2026b). Another direction uses higher-level structure, including entity information and object-centered representations, to preserve important content over time (Liu et al., 2026; Dou et al., 2026). Retrieval-based methods instead keep a broader history and select information relevant to the current generation when it is needed (Ji et al., 2025). These approaches differ both in the evidence used to estimate importance and in whether selection occurs when information is stored or when stored history is later accessed.

KV-cache management in language models. Language models face the same budget with long contexts, and their KV-cache policies keep attention sinks with a recent window (Xiao et al., 2024), evict tokens by accumulated attention (Zhang et al., 2023; Liu et al., 2023), choose eviction policies per attention head (Ge et al., 2024), or select tokens by observed or query-dependent attention (Li et al., 2024; Tang et al., 2024). These policies score tokens by the attention they receive, whereas EncoreMem keeps or releases a stored region by whether later frames re-encounter its content.

D Limitations

EncoreMem manages the cache of a frozen generator and does not change its weights, so it cannot correct errors that the generator makes within a chunk; it only decides which generated content later chunks read. Its decisions use tensors of the generator, so its behavior depends on how that generator

Table D: **Change along the rollout.** Last ten seconds minus first ten seconds of each dimension, computed from the subclip scores of Tables 1 and 4.

	Multi-prompt			Single-prompt		
Method	Subject	Background	Aesthetic	Subject	Background	Aesthetic
Self Forcing	−0.34	−0.17	−7.80	+0.86	+1.02	−9.89
MemFlow	−1.51	−1.58	−2.58	−0.56	−0.44	−0.60
LongLive	−2.18	−2.21	−2.51	−0.79	−0.39	−0.51
Deep Forcing	−2.48	−2.58	−2.03	−0.60	−0.54	−2.28
EncoreMem	−0.94	−1.36	−0.23	+0.46	+0.20	−0.81

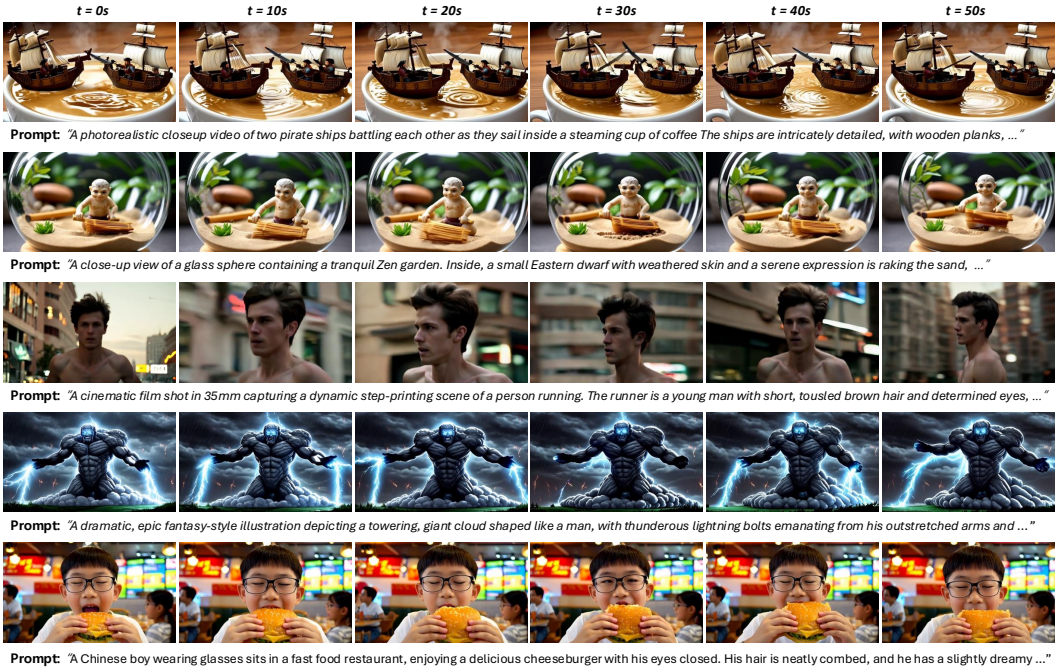

Figure A: **Additional qualitative results with single-prompts.** Frames at 0, 10, 20, 30, 40, and 50 seconds of single-prompt rollouts of EncoreMem; the prompt is shown below each row.

represents content, and we evaluate it on the Self Forcing generator. We measure sixty-second rollouts, the length used by the benchmarks we follow, and show longer rollouts only qualitatively (Appendix E.2).

E Additional Results

E.1 Change Along the Rollout

Tables 1 and 4 average each dimension over the whole rollout, so they do not show whether consistency and visual quality hold up as the video grows. Table D reports, for the same videos and scores, the value over the last ten seconds minus the value over the first ten seconds, obtained by grouping the two-second subclips of the evaluation into ten-second intervals.

E.2 Additional Qualitative Results

Figure A shows further single-prompt rollouts of EncoreMem, in addition to Figure 5. Figures B and C show further multi-prompt rollouts of EncoreMem, on multi-prompts from LongLive (Yang et al., 2026) and on multi-prompts whose segments change gradually, in addition to Figure 1.

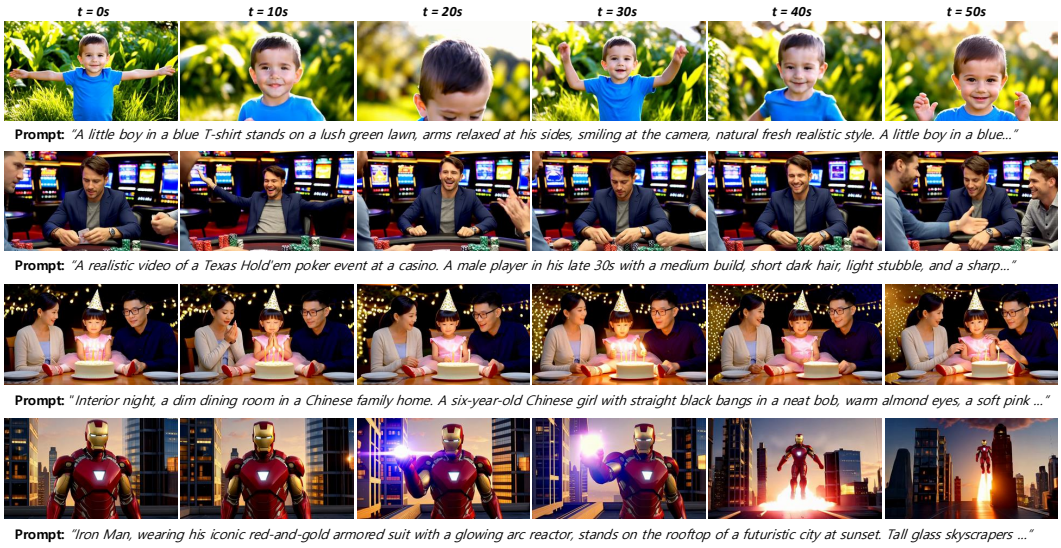

Figure B: **Additional qualitative results on multi-prompts from LongLive.** Frames at 0, 10, 20, 30, 40, and 50 seconds of multi-prompt rollouts of EncoreMem; the prompt is shown below each row.

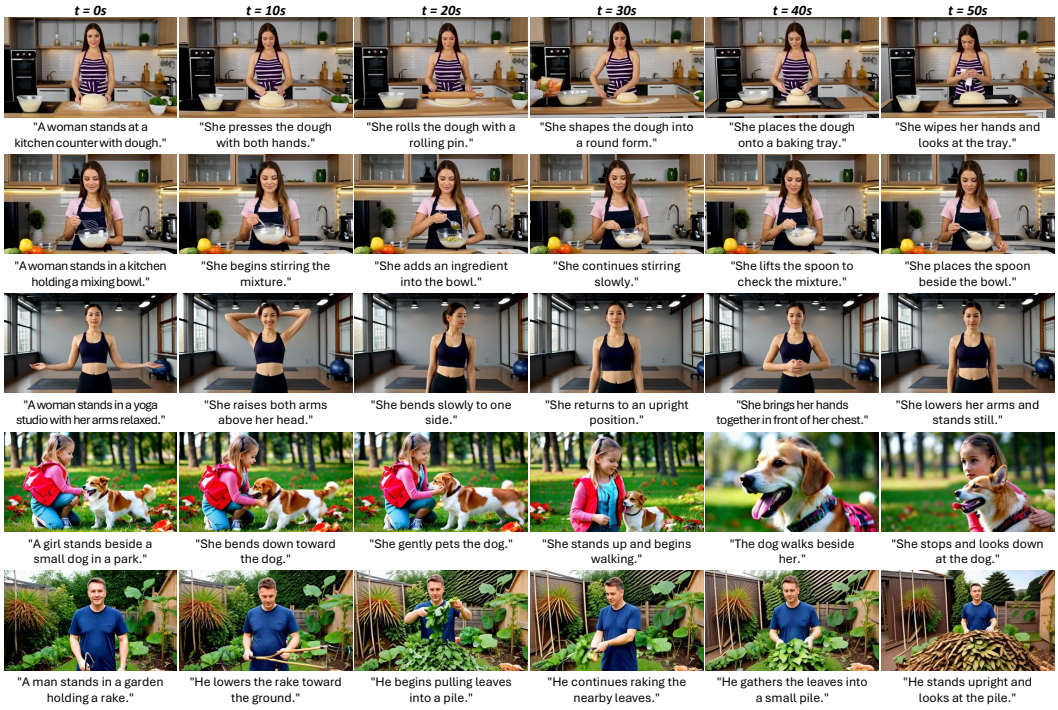

Figure C: **Additional qualitative results on multi-prompts with gradual prompt changes.** Frames at 0, 10, 20, 30, 40, and 50 seconds of multi-prompt rollouts of EncoreMem, with the prompt of each segment below it.

Figure D further examines EncoreMem on 120-second single-prompt rollouts, extending beyond the 60-second evaluation setting used in the main experiments. In these rollouts, the main subjects stay recognizable and artifacts do not visibly accumulate, and in several of them the viewpoint and scene composition change over time.

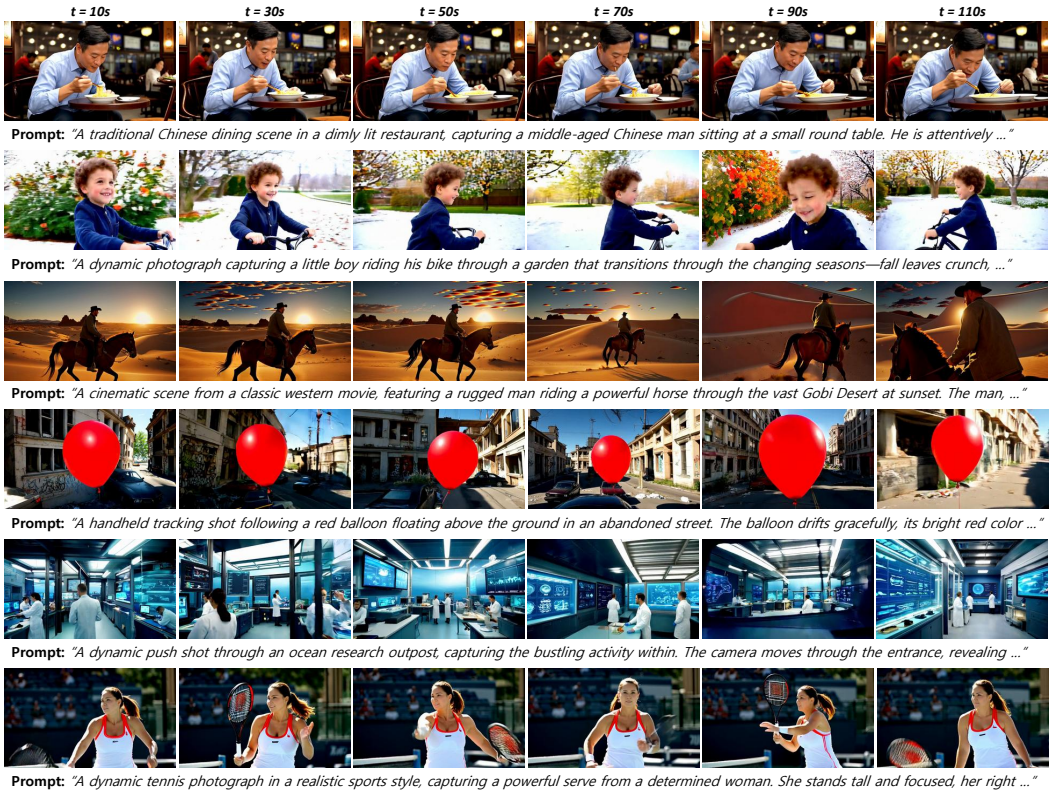

Figure D: **Additional qualitative results for 120-second single-prompt rollouts.** Frames at 10, 30, 50, 70, 90, and 110 seconds. The main subjects stay recognizable over the extended rollouts, and in several of them the viewpoint and scene composition change.