Advancing Narrative Long Video Generation via Training-Free Identity-Aware Memory

Jinzhuo Liu[1], **Jiangning Zhang**[1]†, **Wencan Jiang**[1], **Yabiao Wang**[1,2]
Dingkang Liang[3], **Zhucun Xue**[1], **Ran Yi**[4], **Yong Liu**[1]
[1]Zhejiang University [2]Tencent Youtu Lab
[3]Huazhong University of Science and Technology
[4]Shanghai Jiao Tong University

Abstract

Autoregressive video generation has improved rapidly in visual fidelity and interactivity, but it still suffers from long-term inconsistency and memory degradation. Most existing solutions either compress historical frames using predefined strategies or retrieve keyframes based on coarse implicit attention signals, both of which fail to handle evolving prompts with shifting entity references, leading to identity drift, character duplication, and attribute loss. To address this, we propose **IAMFlow**, a training-free identity-aware memory framework that explicitly models and tracks persistent entity identities, enabling consistent generation across prompt transitions. Specifically, an LLM extracts entities with visual attributes from each prompt and assigns unique global IDs for identity-aware memory, while a VLM asynchronously verifies and refines attributes from rendered frames, enabling explicit entity tracking in place of implicit similarity-based matching. To keep the proposed framework computationally practical, we design a systematic inference acceleration pipeline, including asynchronous visual verification, adaptive prompt transition, and model quantization, which achieves faster generation than existing baselines. Furthermore, we introduce **NarraStream-Bench**, a benchmark for narrative streaming video generation that features 324 multi-prompt scripts spanning six dimensions and a three-dimensional evaluation protocol that integrates both traditional metrics and multimodal large language model-based assessments. Extensive experiments show that IAMFlow, despite being training-free, achieves the best overall performance on NarraStream-Bench, outperforming the strongest baseline by **2.56** points, while achieving a **1.39**× speedup over the most efficient baseline in the 60-second multi-prompt setting.

1 Introduction

Recent advances in video generation have been largely driven by Diffusion Transformer (DiT) models [1–3] with bidirectional attention. However, the quadratic cost of full attention [4] limits these models to generate short videos in a single pass, motivating autoregressive (AR) video generation [5–7] as a more scalable paradigm for long and interactive video synthesis. Despite this promise, the AR paradigm still faces three core issues: (i) *limited historical memory* [8–10], where causal attention is constrained by a local context window and gradually forgets earlier entities and attributes; (ii) *interactive prompt adaptation* [11, 8], where users expect to guide evolving generation through streaming inputs; (iii) *inefficient context scaling* [12, 13], where simply enlarging the local window neither preserves critical memory nor exploits the computational advantage of AR models over bidirectional ones. These failures reveal the same problem: the model lacks an explicit state variable

†Corresponding author: 186368@zju.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

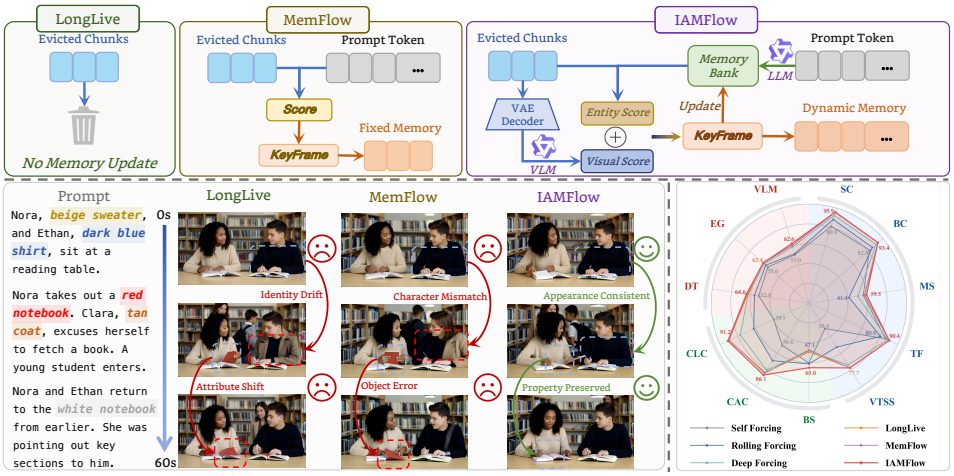

Figure 1: **Top**: Comparison of three representative memory paradigms. **Bottom**: Under evolving narrative prompts, LongLive and MemFlow drift, while IAMFlow preserves entity identities and attributes over 60 seconds and achieves a higher overall score across 11 metrics (see the radar chart).

for persistent entities, so visual evidence stored as positions, frames, or generic features cannot reliably bind a character to its evolving attributes across prompt switches [14].

As illustrated in Fig. 1, existing methods can be viewed through two representative memory paradigms. ***LongLive*** [8], inspired by StreamingLLM [15], mitigates quality degradation in streaming rollouts by introducing sink tokens: it always preserves the first chunk as persistent anchors and adopts streaming training to stabilize autoregressive generation. This design improves long-term consistency, but memory fixed to the first chunk discards information accumulated during generation, which can cause memory loss and visual degradation. This reveals the first challenge: long memory should dynamically preserve critical information rather than rely on a fixed global anchor. ***MemFlow*** [9] introduces explicit memory frames within a dynamic memory bank, retrieving and updating semantically relevant historical frames according to the current prompt. While this enables more adaptive historical recall, it mainly treats memory as a streaming retrieval component without systematic organization. Moreover, increasing the memory or context window can introduce additional inference overhead. This reveals the second challenge: a practical streaming memory framework must maintain a balance between information preservation and efficient AR generation.

Building on these observations, we propose ***IAMFlow***, a training-free framework for narrative autoregressive video generation. To address identity forgetting and memory loss, we organize an identity-aware memory to introduce explicit entity management into the autoregressive generation process, enabling the model to recall persistent entities and attributes across prompt transitions. To address the efficiency challenge, we further adopt a systematic inference acceleration pipeline that makes the proposed memory framework computationally practical. Since existing benchmarks do not fully cover streaming multi-prompt narrative scenarios, we also introduce ***NarraStream-Bench***, a benchmark designed to address these evaluation gaps. The framework, acceleration design, benchmark construction, and evaluation protocol are presented in Sec. 3.1, Sec. 3.2, and Sec. 3.3.

In summary, our contributions are as follows:

- We introduce **IAMFlow**, a training-free identity-aware memory framework that explicitly organizes historical information around persistent entities and attributes, enabling reliable identity preservation across evolving prompt transitions.
- We design a systematic inference acceleration pipeline to make the framework computationally practical, combining asynchronous visual verification, adaptive prompt transition, and model quantization to preserve long-term consistency without sacrificing generation speed.
- We introduce **NarraStream-Bench**, a modern benchmark suite for assessing long-term consistency in narrative streaming video generation. Extensive experiments and ablation studies demonstrate that IAMFlow achieves superior performance across various metrics while enabling more efficient inference.

2 Related Work

Autoregressive Video Generation Bidirectional video diffusion models [2, 3] are constrained to one-shot, non-interactive short video generation, whereas autoregressive (AR) video models predict future frames conditioned on previous context. Early methods [16, 6] adopt Teacher Forcing (TF) for training, using ground-truth historical frames as conditioning input, which suffers from a severe train-test gap during long-horizon rollout. Subsequent systems [17, 18] adopt Diffusion Forcing (DF) [5], which injects noise into the conditioning context during training to mitigate error accumulation. Building on this foundation, recent works further explore causal modeling and efficient generation. CausVid [7] distills a pretrained bidirectional model into a few-step causal generator. Self Forcing [19] simulates autoregressive rollouts during training, reducing the train-test discrepancy and error propagation. Follow-up forcing-style methods [20–23] improve robustness to long-horizon rollout. In addition, PA-VDM [24] introduces progressive frame-wise noise scheduling, while StreamingT2V [11] introduces short-term and long-term modules for extendable generation.

Memory Mechanisms in Video Generation A central challenge in autoregressive long video generation is preserving relevant history over long horizons under bounded computation and memory. Existing methods address this problem through three main memory designs. Compressed-context approaches reduce the growing video history into a compact conditioning signal: FramePack [10] packs historical frames into a fixed context budget, FAR [25] combines fine recent context with coarser distant history, and LoViC [13] and PFP [26] compress long histories into latent or retrievable memory representations. Parameterized memory methods instead store history in learnable internal states. TTT-Video [27] updates sequence-specific fast weights at test time, SlowFast-VGen [28] stores episodic information in inference-time temporal LoRA parameters, and VideoSSM [29] combines local autoregressive context with state-space memory. Retrieval-based methods select relevant history during generation. MemoryPack-based generation [30] retrieves and fuses text-visual history, while InfLVG [31] learns inference-time context selection under a fixed memory budget. LongLive [8] maintains a streaming cache with frame-level attention sinks and KV-recache, and MemFlow [9] retrieves semantically relevant memory frames from a dynamic memory bank.

Benchmarks for Video Generation Existing video-generation benchmarks [32–34], along with recent narrative long-video evaluations [35, 36], cover many aspects of long-horizon video generation, but they do not fully cover streaming multi-prompt narrative scenarios. VBench and VBench++ mainly provide broad quality and semantic metrics for general video generation, while VBench-Long evaluates temporal consistency within a single long-form generation by splitting videos into semantically consistent clips and using a slow-fast protocol. BlockVid focuses on minute-long video generation, and MovieBench and NarrLV move evaluation toward movie-level and narrative settings. These benchmarks do not explicitly test continual adaptation to evolving prompts, instruction following under state changes between segments, or long-range entity recall after reappearance.

3 Method

We present IAMFlow, a training-free identity-aware memory framework that builds an explicit ID-centric memory bank to improve temporal consistency in long video generation (Sec. 3.1). To make the proposed memory framework computationally practical, we further develop a systematic inference acceleration pipeline for efficient streaming generation (Sec. 3.2). To evaluate the proposed framework, we introduce NarraStream-Bench, a benchmark suite for assessing long-term consistency in narrative streaming video generation (Sec. 3.3).

Formulation of Memory-aware Autoregressive Video Generation. Given a stream of prompts $\{p_t\}_{t=1}^{T}$ and corresponding generated video units $\{x_t\}_{t=1}^{T}$, where each x_t can denote a frame or latent chunk, memory-aware autoregressive video diffusion models factorize generation as

$$p_\theta(x_{1:T} \mid p_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid \mathcal{L}_t, \mathcal{M}_t, p_t)\,. \tag{1}$$

Here p_t is the active prompt for generation unit x_t, $\mathcal{L}_t$ is the recent local context, instantiated as $x_{t-W_{\text{loc}}+1:t-1}$ with local window size W_{loc}, and $\mathcal{M}_t$ is the compact long-term memory selected from a method-specific memory state $\mathcal{S}_t$ under a budget B. Under this formulation, existing methods can be organized into a **Memory Organization Paradigm**, where they share the same autoregressive objective in Eq. 1 but differ fundamentally in how they represent and update $\mathcal{M}_t$:

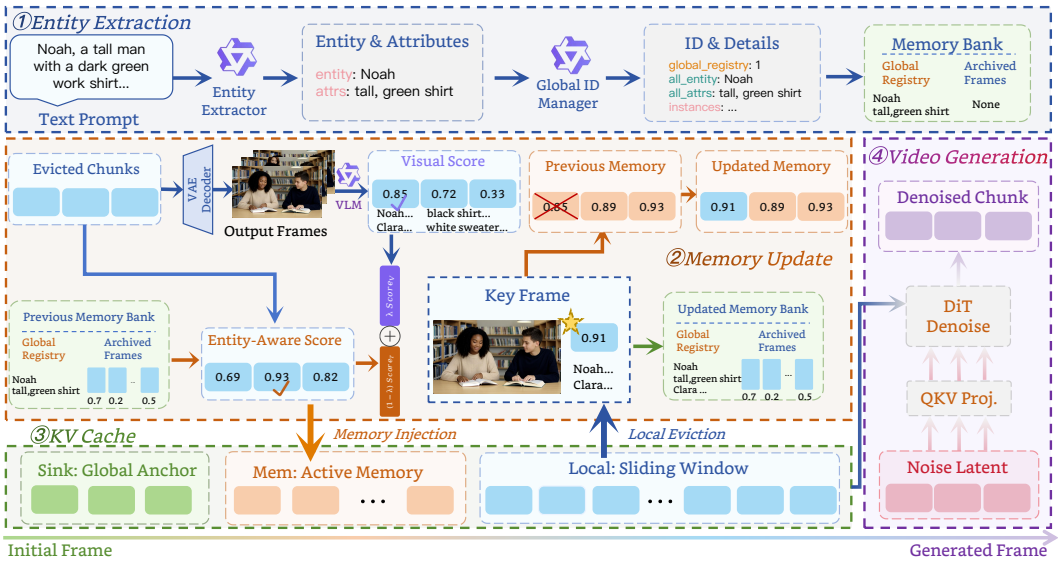

Figure 2: **Framework of IAMFlow**. (1) An LLM extracts entities and attributes from each prompt and assigns persistent global IDs to build an ID-centric memory bank. (2) At each chunk generation, the memory bank retrieves frames associated with critical entities. (3) During autoregressive generation, evicted local chunks are decoded and scored, while asynchronous VLM verification updates attributes and selects reliable keyframes. (4) The latent noise is concatenated with the organized KV cache and fed into the DiT to generate the denoised chunk.

Positional Anchoring. Early approaches such as LongLive [8] instantiate $\mathcal{M}_t$ with persistent sink tokens anchored solely to the initial temporal positions:

$$p_{\mathrm{LL}}(x_{1:T} \mid p_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{t-W_{\mathrm{loc}}+1:t-1},\, \mathrm{Sink}(x_{1:s}),\, p_t)\,. \tag{2}$$

While mitigating degradation in early generation steps, this static strategy fails to capture the dynamic evolution of narrative events beyond the first chunk.

Token-Level Retrieval. To enable dynamic updates, MemFlow [9] uses a dynamic memory bank, organizing $\mathcal{M}_t$ based on token-level relevance retrieved via the current prompt:

$$p_{\mathrm{MF}}(x_{1:T} \mid p_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{t-W_{\mathrm{loc}}+1:t-1},\, \mathrm{Retrieve}(\mathcal{F}_t, p_t),\, p_t)\,. \tag{3}$$

Although this design supports adaptive memory updates through $\mathcal{F}_t$, it still operates on frame-level features rather than explicit entity identities. This makes it prone to *feature conflation*: different entities with similar visual or semantic attributes may be confused, while less prominent subjects may not be reliably retrieved when targeted recall is needed.

3.1 Identity-Aware Memory Preservation

To address the limitations above, IAMFlow elevates memory organization from low-level tokens to high-level symbolic entities. Specifically, we instantiate $\mathcal{M}_t$ as an ID-centric memory state, which enables targeted recall of persistent entities and reduces identity conflation in previous memory designs.

We propose an **Identity-Aware Memory** mechanism (Fig. 2) that maintains a memory bank, including a global entity registry $\mathcal{R}$, a frame archive $\mathcal{F}$, and an active memory buffer m^{id}. At each prompt p_t, the mechanism executes four phases: **(i)** an LLM [37] extracts entities and assigns persistent global IDs; **(ii)** the memory bank retrieves frames covering all active identities; **(iii)** during generation, evicted chunks are scored and the best frame is archived; **(iv)** a VLM [38] asynchronously scores

visual quality and corrects attribute drift. IAMFlow therefore instantiates Eq. 1 as:

$$p_{\mathrm{IAM}}(x_{1:T} \mid p_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t \mid x_{t-W_{\mathrm{loc}}+1:t-1},\, \mathrm{IDRetrieve}(\mathcal{F}_t, \mathcal{E}_t, \mathcal{R}_t),\, p_t)\,. \tag{4}$$

Here, $\mathcal{R}_t$ and $\mathcal{F}_t$ together define the IAMFlow identity-aware memory state, while $\mathcal{E}_t$ denotes the entity descriptors parsed from the current prompt and aligned with the global registry before identity-aware retrieval.

Entity Extraction and ID Assignment. At the first chunk of each prompt p_t, an LLM parses the text into structured entity descriptors $\mathcal{E}_t = \{(e_j, \mathbf{a}_j)\}$, where e_j is an entity name and $\mathbf{a}_j$ lists its stable visual attributes. Each entity is linked to a persistent ID in the global registry $\mathcal{R} = \{g_k : (\mathrm{name}_k, \mathbf{A}_k)\}$ via a matching procedure for its novelty: first, explicit novelty markers trigger immediate allocation of a new ID. Otherwise, the LLM compares the current attribute set $\mathbf{a}_j$ with the registered attribute sets $\{\mathbf{A}_k\}$ to resolve ambiguous references. If a confident match is found, the entity is linked to the corresponding existing ID; if not, a new ID is allocated. The registry is then updated with the new observation and stored in the memory bank.

Identity-aware Frame Scoring, Archival, and Retrieval. The memory bank maintains a growing frame archive $\mathcal{F}$. During chunk-wise generation, when the oldest chunk is evicted from the local window, we score its frames and archive the best one. To prioritize content tied to active identities, we construct an entity-token weight vector $\boldsymbol{\omega} \in \mathbb{R}^S$ over the prompt tokens according to the entity names and attributes maintained in the memory bank, and normalize it as $\tilde{\omega}_u = \omega_u / \sum_{v=1}^{S} \omega_v$. For layer $l \in \mathcal{L}$ and attention head $h \in \mathcal{H}$, we aggregate the cached text keys and the visual keys for frame f as $\bar{\mathbf{r}}_{\mathrm{id}}^{l,h} = \sum_{u=1}^{S} \tilde{\omega}_u \mathbf{K}_{u,h}^{l,\mathrm{text}}$ and $\bar{\mathbf{k}}_f^{l,h} = \frac{1}{n_f} \sum_{v=1}^{n_f} \mathbf{K}_{f,v,h}^{l,\mathrm{vis}}$. The entity score is computed by

$$s_{\mathrm{entity}}(f) = \sum_{l \in \mathcal{L}} \beta_l \frac{1}{H} \sum_{h=1}^{H} \frac{\langle \bar{\mathbf{r}}_{\mathrm{id}}^{l,h}, \bar{\mathbf{k}}_f^{l,h} \rangle}{\sqrt{d}}. \tag{5}$$

This score reuses the generator's internal compatibility between text-conditioned keys and visual keys as a lightweight saliency proxy for ranking memory frames that best preserve identity. The entity score is then fused with the VLM-based score as $s(f) = (1-\lambda)\, s_{\mathrm{entity}}(f) + \lambda\, s_{\mathrm{visual}}(f)$. The best frame is stored in $\mathcal{F}$ along with its entity annotations and KV cache. We formulate the ID retrieval as a maximum coverage problem under a memory budget. Exploiting the inherent submodularity of this NP-hard problem, we adopt a greedy approximation. Each archived frame f is associated with an entity set $\mathcal{G}(f)$ and a score, and we iteratively select the frame with the highest score until all current IDs $\{g_j\}$ are covered. The KV caches of the selected frames are then assembled into the active memory m_t^{id} and injected into the model.

Asynchronous Visual Verification and Correction. Following each chunk, the denoised latent is asynchronously decoded into pixels via the VAE. Then a VLM scores each decoded frame for visual fidelity to the target entities and returns $s_{\mathrm{visual}}(f) \in [0, 1]$ for the fusion score above. For the first chunk of each prompt, the VLM also verifies entity attributes from the rendered pixels and writes corrected observations back to the global registry, thereby reducing error accumulation across prompt switches. Detailed runtime implementation and deterministic prompt templates are provided in Appendices A and C.

3.2 Systematic Inference Acceleration

Prior methods in autoregressive long video generation are bottlenecked by two recurring costs: full KV cache recomputation at every prompt switch and synchronous frame decoding [39]. We introduce three complementary strategies that eliminate both bottlenecks, enabling IAMFlow to achieve faster end-to-end inference than prior methods even with the additional components.

Adaptive Prompt Transition. In prior autoregressive generation methods [8, 9], each prompt switch requires full recaching: all context frames must be re-encoded under the new text condition before generation can resume. We introduce an **Adaptive Prompt Transition** (APT) strategy that eliminates this cost by smoothing the prompt switch inside the cross-attention key-value conditioning space. We view APT as smoothing the text condition across segment boundaries. Instead of switching abruptly from one prompt to the next, APT interpolates the cross-attention keys and values used by

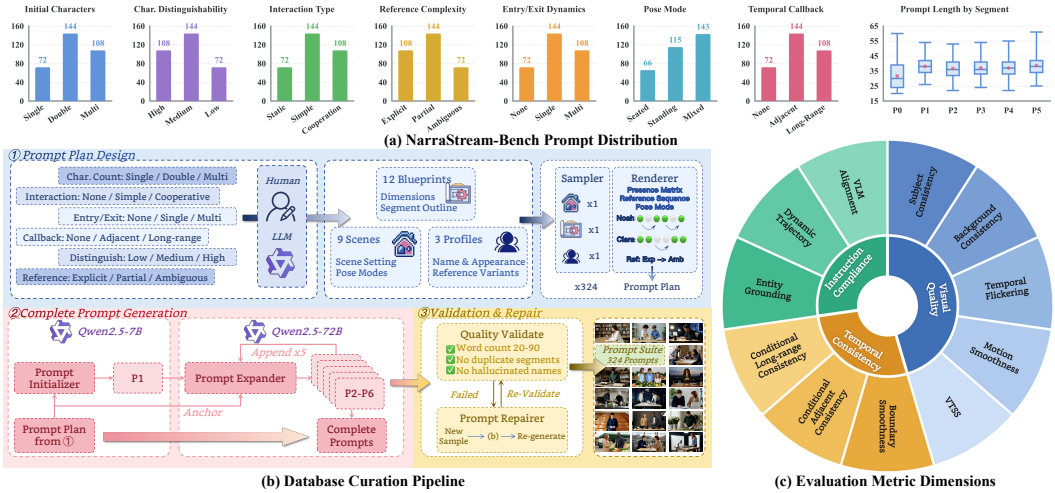

Figure 3: **Overview of NarraStream-Bench**.

the DiT, so the denoiser receives a gradual change in conditioning. This leads to smoother denoising directions across neighboring chunks. At the switch point, the current cross-attention keys and values $(\mathbf{K}_{\text{old}}, \mathbf{V}_{\text{old}})$ are cached, and subsequent chunks use blended states:

$$\mathbf{K} = (1-\alpha)\,\mathbf{K}_{\text{old}} + \alpha\,\mathbf{K}_{\text{new}}, \quad \mathbf{V} = (1-\alpha)\,\mathbf{V}_{\text{old}} + \alpha\,\mathbf{V}_{\text{new}}, \tag{6}$$

where α increases from 0 to 1 over a transition window following a cosine schedule $\alpha(\tau) = \frac{1}{2}(1 - \cos \pi\tau)$. To adapt the transition length to the magnitude of the prompt change, we compute the cosine distance d between the averaged prompt embeddings of two adjacent segments, reusing the existing outputs of the text encoder at no additional cost, and set the transition window length to $W_{\text{apt}} = W_{\min} + d \cdot (W_{\max} - W_{\min})$, rounded to the nearest chunk boundary. As a result, minor prompt edits use shorter transitions, while larger narrative changes trigger longer blending windows. By introducing prompt changes gradually over a short temporal window, APT reduces boundary flicker, identity drift, and transition artifacts. It also avoids the full recache pass required by hard prompt switching, replacing each switch with a lightweight blend of the corresponding key and value.

Asynchronous Verification Pipeline. Rather than interrupting autoregressive generation, we integrate the VAE decoding and VLM scoring loop with the local window eviction mechanism. Once a mature chunk is evicted, a background thread decodes it and runs VLM verification in parallel with DiT denoising for subsequent chunks. This keeps only visually reliable, identity-consistent frames in long-term memory while overlapping verification with generation.

Efficient Deployment. To further accelerate autoregressive generation with identity-aware memory, we deploy the DiT, VAE, and text encoder on one GPU, while serving the LLM and VLM on another GPU via vLLM [40]. We further replace the default Wan VAE [2] with a Wan2.1-adapted TinyVAE based on Tiny AutoEncoder [41], and quantize all DiT linear layers to FP8 [42] following LightX2V [43]. These optimizations provide additional speedup.

3.3 NarraStream-Bench: Evaluating Narrative Streaming Video Generation

To address the lack of benchmarks for continual prompt adaptation, instruction compliance between segments, and long-term consistency in streaming narrative video generation, we introduce **NarraStream-Bench**, a structured benchmark comprising 324 narrative scripts, each spanning 60 seconds and divided into six consecutive segments, together with metrics specifically designed for streaming narrative video generation (see Fig. 3). Table 1 compares the evaluation protocols of NarraStream-Bench and related representative benchmarks. We provide the implementation details for database construction and evaluation metrics in Appendices D and E, respectively.

Test Suite Construction. As shown in Fig. 3(b), we define six challenge dimensions for narrative streaming video generation and build a prompt database of scene templates, narrative blueprints, and character profiles through human drafting and LLM refinement. We then sample prompt plans from

Table 1: **Comparison of related long-video generation benchmarks**. Note that the custom VBench-Long protocol only supports metrics for visual quality.

Benchmark	VQ	TC	IC	Prompt Type	Aggregation Strategy	Year
VBench-Long [33]	✓	×	×	Single	Slow-Fast Avg.	2024
LV-Bench [34]	✓	✓	×	Single	VDE	2025
NarrLV [36]	×	✓	✓	Single	TNA-based QA	2025
NarraStream-Bench	✓	✓	✓	Multi	Narrative-Aware	2026



0-10s: **Miles**, a slim man with a ***buzz cut*** and a ***charcoal jacket***, and **Luna**, a petite woman with ***silver hair*** and a ***black turtleneck***, sort papers and cups at a table.
10-20s: **Nora**, a young woman with short ***black curls*** and a ***beige sweater***, enters and approaches the table where they sit. **Nora** hands a small item to **Luna**.
20-30s: **Luna** takes the small item, examining it closely under the bright overhead lighting. **Miles** watches her intently, while **Nora** remains.
30-40s: **Luna** and Miles both lean in to inspect the small item. Luna adjusts it slightly, and **Miles** nods in agreement. **Nora** leaves the space.
40-50s: **Miles** points at the updated arrangement on the table, and Luna nods. They adjust the items with focused precision.
50-60s: **Miles** and **Luna** finish their adjustments and step back. They exchange a final glance.

Figure 4: **Qualitative results of IAMFlow.** Our method keeps identities and attributes consistent across the 60-second video, while other baselines show varying degrees of identity shift and attribute drift.

this database, augment each plan with a presence matrix and reference sequence between segments, and generate six coherent segments following the formats of Wan2.1 and LongLive. After quality validation and repair for logical or formatting issues, the final suite contains 324 prompt sequences.

Evaluation Metrics. We define 11 metrics across three complementary groups, as illustrated in Fig. 3(c): 1) ***Visual Quality*** measures visual fidelity with five metrics: Subject Consistency (SC), Background Consistency (BC), Temporal Flickering (TF), Motion Smoothness (MS), and VTSS; 2) ***Temporal Consistency*** measures transition smoothness and long-term consistency with three metrics: Boundary Smoothness (BS), Conditional Adjacent Consistency (CAC), and Conditional Long-range Consistency (CLC); 3) ***Instruction Compliance*** measures instruction following with three metrics: Entity Grounding (EG), Dynamic Trajectory (DT), and VLM Alignment (VLM) [44, 45]. Rather than averaging scores uniformly, we use an LLM to assign segment weights for each metric based on the prompt sequence. The final score for each group is computed as an importance-weighted average across segments, and the overall score is the arithmetic mean of the three group scores.

4 Experiments

Implementation Details. IAMFlow uses the MemFlow [9] weights to generate 60-second videos from six successive prompts at 832×480 resolution, adding the proposed identity-aware memory and inference pipeline without additional training. The KV cache is organized as $N_s = 3$ sink latent

Table 2: **Quantitative comparison for the multi-prompt 60-second setting on NarraStream-Bench**. Self Forcing, Rolling Forcing, and Deep Forcing are adapted for the experiment. The best results are highlighted in bold, and the second-best results are underlined.

Model	Visual Quality ↑					Temporal Consistency ↑			Instruction Compliance ↑			Overall ↑
	SC	BC	TF	MS	VTSS	BS	CAC	CLC	EG	DT	VLM	
Self Forcing [19]	80.85	82.05	86.29	56.17	30.39	60.25	50.57	39.54	55.63	51.98	50.96	56.71
Rolling Forcing [20]	88.16	85.93	80.55	41.41	52.59	<u>61.19</u>	68.83	62.18	58.45	56.37	52.03	63.14
Deep Forcing [12]	92.27	90.74	88.54	50.61	73.99	48.10	80.09	77.28	60.71	62.87	59.35	69.57
LongLive [8]	95.36	<u>93.32</u>	<u>90.27</u>	<u>57.76</u>	77.29	49.19	83.06	<u>89.97</u>	**62.40**	63.33	<u>62.22</u>	<u>73.17</u>
MemFlow [9]	<u>95.37</u>	93.21	90.23	57.37	**77.67**	47.06	<u>83.92</u>	89.71	61.93	<u>64.14</u>	60.85	72.88
IAMFlow	**95.88**	**93.43**	**90.36**	**59.49**	<u>77.56</u>	**65.03**	**86.07**	**91.25**	<u>62.03</u>	**64.56**	**62.61**	**75.73**



Figure 5: **Left**: Qualitative results of ablation experiments. **Right**: *Top*, the dynamic strategy achieves the best balance between temporal consistency and instruction compliance; *Bottom*, Adaptive Prompt Transition (APT) enables smoother boundary transitions.

frames, an active memory buffer of $N_m \in [2, 4]$ retrieved latent frames, and a sliding local window of $N_l = 9$ latent frames; we deploy Qwen3-4B-Instruct [37] and Qwen3-VL-2B-Instruct [38] as the LLM and VLM via vLLM [40]. Keyframe scoring uses $\lambda = 0.3$, Adaptive Prompt Transition uses $W_{\min} = 3$ and $W_{\max} = 15$ chunks, and all experiments are conducted on NVIDIA H20 GPUs.

Baselines. We benchmark IAMFlow against comparable open-source video generation models, including Self Forcing [19], Rolling Forcing [20], Deep Forcing [12], LongLive [8], and MemFlow [9]. We evaluate all methods on **NarraStream-Bench**, report 11 metrics scaled by 100, and additionally measure human alignment and inference efficiency under the same evaluation budget.

4.1 Comparison on NarraStream-Bench

As several representative baselines were originally designed for single-prompt rollouts, we extend Self Forcing, Rolling Forcing, and Deep Forcing to the streaming video generation setting; Appendix F provides the details. We then compare our method against all baselines. Qualitative and quantitative results are shown in Fig. 4 and Table 2. IAMFlow achieves the best overall score, with the most pronounced gains on the metrics that stress entity-state memory: Boundary Smoothness, Conditional Adjacent Consistency, and Conditional Long-range Consistency. In contrast, LongLive and MemFlow show limited long-term consistency, supporting our central claim that narrative long video generation benefits from remembering persistent entities. The adapted forcing baselines lag further behind, as they mainly preserve recent context and lack a memory mechanism for long video generation. To further validate the evaluation protocol, we examine evaluator robustness with judges from outside the Qwen family in Appendix G. Further qualitative comparisons and discussions of challenging failure cases and limitations are provided in Appendices J and I.

Table 3: **Unified ablation studies of IAMFlow**. The best results are highlighted in bold, and the second-best results are underlined.

Model	Overall ↑	Visual Quality ↑	Temporal Consistency ↑	Instruction Compliance ↑
Memory Mechanism				
w/o VLM verification	<u>75.41</u>	83.17	<u>80.22</u>	62.83
w/o LLM planning	75.26	83.09	80.12	62.58
w/o ID memory	74.28	82.74	77.85	62.25
Memory Retrieval				
Random	73.10	82.39	73.89	<u>63.02</u>
Visual-Only	71.35	80.94	71.08	62.04
Semantic-Only	73.49	82.70	74.85	62.91
Memory Allocation				
Fixed 3	73.69	82.71	75.53	62.84
Fixed 6	74.00	<u>83.24</u>	76.72	62.05
Prompt Transition				
Full KV recache	75.27	83.12	79.71	62.97
No transition	75.10	83.01	79.42	62.88
Ours	**75.73**	**83.34**	**80.78**	**63.07**

Table 4: **End-to-end efficiency comparison**. IAMFlow achieves the highest speedup in the 60-second multi-prompt setting across representative baselines.

Method	Training-free	E2E Latency (s) ↓	Speedup ↑	FPS ↑
Self Forcing	×	264.86	1.0×	4.72
Rolling Forcing	×	312.11	0.85×	4.03
Deep Forcing	✓	250.62	1.06×	5.11
LongLive	×	<u>218.43</u>	<u>1.21×</u>	<u>6.14</u>
MemFlow	×	241.76	1.10×	5.36
IAMFlow	✓	**156.69**	**1.69×**	**6.20**

4.2 Ablation Studies

We evaluate four components in IAMFlow: **Memory Mechanism**, **Memory Retrieval**, **Memory Allocation**, and **Prompt Transition**. Table 3 and Fig. 5 show that the full IAMFlow achieves the best overall performance. Removing the identity-aware memory causes the largest degradation, confirming that explicit long-term memory is critical to identity preservation and temporal consistency. LLM planning and VLM verification provide complementary gains through structured entity decomposition and visual grounding. Identity-aware retrieval outperforms random, visual-only, and semantic-only retrieval, while dynamic allocation and Adaptive Prompt Transition further improve the balance between identity coverage, instruction compliance, and transition smoothness.

4.3 Efficiency Comparison

We incorporate a systematic inference acceleration pipeline into IAMFlow, enabling faster end-to-end inference than prior methods despite the additional memory components. As shown in Table 4, results on H20 GPUs show that our method achieves a $1.69\times$ end-to-end speedup compared to Self Forcing in the same multi-prompt setting. This comparison reflects the overall efficiency of our method. Appendix B further decomposes the contributions of the pipeline components. Notably, even with the additional memory module, IAMFlow still delivers slightly higher FPS than LongLive.

4.4 Metrics on VBench-Long

We also evaluate IAMFlow on VBench-Long [33], a benchmark for long video generation. Since VBench-Long is designed mainly for single-prompt evaluation, we follow LongLive [8] and MemFlow [9] and use the custom-prompt protocol, with prompts drawn from the NarraStream-Bench database. Under this protocol, VBench-Long supports six metrics: Subject Consistency, Background Consistency, Temporal Flickering, Motion Smoothness, Overall Consistency, and Dynamic Degree. As shown in Table 5, IAMFlow achieves the best performance on five of the six metrics and remains competitive on Motion Smoothness.

Table 5: **Evaluation metrics on VBench-Long**. All scores are multiplied by 100. Best results are highlighted in bold, and second-best results are underlined.

Method	Subject Consistency ↑	Background Consistency ↑	Temporal Flickering ↑	Motion Smoothness ↑	Overall Consistency ↑	Dynamic Degree ↑
LongLive	<u>98.956</u>	<u>96.394</u>	<u>99.402</u>	**17.202**	<u>59.420</u>	<u>75.895</u>
MemFlow	98.946	96.170	99.386	16.440	57.667	75.726
IAMFlow	**99.006**	**96.554**	**99.438**	<u>16.522</u>	**59.630**	**75.942**

4.5 Human Alignment

To validate whether our evaluation metrics align with human perception, we conduct a user study on 30 samples generated by LongLive, MemFlow, and IAMFlow. We recruit 30 volunteers to rank the three model outputs for each metric, convert the rankings into pairwise preferences, and compute Spearman's rank correlation [46] between human preferences and automatic score margins. As shown in Table 6, NarraStream-Bench metrics are highly consistent with human judgments. For each case, every volunteer viewed the complete six-prompt narrative and ranked the three anonymized videos, presented in shuffled order. Ties were allowed. For each of the 11 metrics, we average pairwise preferences over annotators and compute correlation across 90 sample–model-pair comparisons. The full human-study protocol is provided in Appendix H.

Table 6: **Spearman's ρ between human preferences and automatic scores.** These scores show that NarraStream-Bench metrics are highly aligned with human judgments.

	Visual Quality					Temporal Consistency			Instruction Compliance		
	SC	BC	TF	MS	VTSS	BS	CAC	CLC	EG	DT	VLM
ρ	0.8936	0.9103	0.8842	0.8774	0.8985	0.8210	0.8316	0.8834	0.8104	0.9053	0.8671

5 Conclusion

We present **IAMFlow**, a training-free framework for narrative streaming video generation that reframes long-term consistency as identity-aware memory preservation rather than generic context extension. By organizing historical information around persistent entity IDs and visual attributes, IAMFlow enables streaming video generation to preserve identities across evolving multi-prompt narrative scenarios, where prior memory designs often suffer from identity drift, character duplication, or attribute loss. To make the proposed framework computationally practical, IAMFlow further integrates a systematic inference acceleration pipeline, including asynchronous visual verification, adaptive prompt transition, and model quantization. We also introduce **NarraStream-Bench**, a modern benchmark for multi-prompt narrative streaming video generation. Extensive experiments show that IAMFlow achieves the best overall performance on NarraStream-Bench.

Limitations and Future Work. IAMFlow prioritizes persistent identity consistency, which can make it conservative when prompts demand abrupt semantic changes, complex actions, or fine-grained relational updates. This reflects a broader trade-off between memory stability and prompt responsiveness in narrative streaming video generation. Future work will explore adaptive memory and transition strategies that maintain this balance across prompt switches, as well as more flexible entity representations that can capture evolving attributes and relationships without losing identity coherence.

Acknowledgements and Disclosures

This work was supported by the National Key R&D Program of China (No. 2025YFF0511302). The authors declare no competing interests.

References

[1] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[2] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[3] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2025.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[5] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[6] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22963–22974, 2025.

[8] 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.

[9] 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.

[10] Lvmin Zhang and Maneesh Agrawala. Packing input frame context in next-frame prediction models for video generation. *CoRR*, abs/2504.12626, 2025.

[11] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2568–2577, 2025.

[12] 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. *arXiv preprint arXiv:2512.05081*, 2025.

[13] Jiaxiu Jiang, Wenbo Li, Jingjing Ren, Yuping Qiu, Yong Guo, Xiaogang Xu, Han Wu, and Wangmeng Zuo. Lovic: Efficient long video generation with context compression. *arXiv preprint arXiv:2507.12952*, 2025.

[14] Mohamed Elmoghany, Ryan Rossi, Seunghyun Yoon, Subhojyoti Mukherjee, Eslam Mohamed Bakr, Puneet Mathur, Gang Wu, Viet Dac Lai, Nedim Lipka, Ruiyi Zhang, et al. A survey on long-video storytelling generation: architectures, consistency, and cinematic quality. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7023–7035, 2025.

[15] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[16] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong MU, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[17] Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

[18] Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

[19] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

[20] Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025.

[21] Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[22] Yunhong Lu, Yanhong Zeng, Haobo Li, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jiapeng Zhu, Hengyuan Cao, Zhipeng Zhang, Xing Zhu, et al. Reward forcing: Efficient streaming video generation with rewarded distribution matching distillation. *arXiv preprint arXiv:2512.04678*, 2025.

[23] Shuo Chen, Cong Wei, Sun Sun, Ping Nie, Kai Zhou, Ge Zhang, Ming-Hsuan Yang, and Wenhu Chen. Context forcing: Consistent autoregressive video generation with long context. *arXiv preprint arXiv:2602.06028*, 2026.

[24] Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6322–6332, 2025.

[25] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025.

[26] Lvmin Zhang, Shengqu Cai, Muyang Li, Chong Zeng, Beijia Lu, Anyi Rao, Song Han, Gordon Wetzstein, and Maneesh Agrawala. Pretraining frame preservation in autoregressive video memory compression. *arXiv preprint arXiv:2512.23851*, 2025.

[27] Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17702–17711, 2025.

[28] Yining Hong, Beide Liu, Maxine Wu, Yuanhao Zhai, Kai-Wei Chang, Linjie Li, Kevin Lin, Chung-Ching Lin, Jianfeng Wang, Zhengyuan Yang, et al. Slowfast-vgen: Slow-fast learning for action-driven long video generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[29] Yifei Yu, Xiaoshan Wu, Xinting Hu, Tao Hu, Yangtian Sun, Xiaoyang Lyu, Bo Wang, Lin Ma, Yuewen Ma, Zhongrui Wang, et al. Videossm: Autoregressive long video generation with hybrid state-space memory. *arXiv preprint arXiv:2512.04519*, 2025.

[30] Xiaofei Wu, Guozhen Zhang, Zhiyong Xu, Yuan Zhou, Qinglin Lu, and Xuming He. Pack and force your memory: Long-form and consistent video generation. *arXiv preprint arXiv:2510.01784*, 2025.

[31] Xueji Fang, Liyuan Ma, Zhiyang Chen, Mingyuan Zhou, and Guo-jun Qi. Inflvg: Reinforce inference-time consistent long video generation with grpo. *arXiv preprint arXiv:2505.17574*, 2025.

[32] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

[33] Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, et al. Vbench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[34] Zeyu Zhang, Shuning Chang, Yuanyu He, Yizeng Han, Jiasheng Tang, Fan Wang, and Bohan Zhuang. Blockvid: Block diffusion for high-quality and consistent minute-long video generation. *arXiv preprint arXiv:2511.22973*, 2025.

[35] Weijia Wu, Mingyu Liu, Zeyu Zhu, Xi Xia, Haoen Feng, Wen Wang, Kevin Qinghong Lin, Chunhua Shen, and Mike Zheng Shou. Moviebench: A hierarchical movie level dataset for long video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28984–28994, 2025.

[36] Xiaokun Feng, Haiming Yu, Meiqi Wu, Shiyu Hu, Jintao Chen, Chen Zhu, Jiahong Wu, Xiangxiang Chu, and Kaiqi Huang. Narrlv: Towards a comprehensive narrative-centric evaluation for long video generation. *arXiv preprint arXiv:2507.11245*, 2025.

[37] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[38] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[39] Haocheng Xi, Shuo Yang, Yilong Zhao, Muyang Li, Han Cai, Xingyang Li, Yujun Lin, Zhuoyang Zhang, Jintao Zhang, Xiuyu Li, et al. Quant videogen: Auto-regressive long video generation via 2-bit kv-cache quantization. *arXiv preprint arXiv:2602.02958*, 2026.

[40] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[41] Ollin Boer Bohan. Seraena: Wip pytorch code for stably training single-step, mode-dropping, deterministic autoencoders. https://github.com/madebyollin/seraena, 2024.

[42] Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022.

[43] LightX2V Contributors. Lightx2v: Light video generation inference framework. https://github.com/ModelTC/LightX2V, 2025.

[44] Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, et al. Videoscore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2105–2123, 2024.

[45] Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8406–8416, 2025.

[46] C Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

[47] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[49] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[50] Zhen Li, Zuo-Liang Zhu, Ling-Hao Han, Qibin Hou, Chun-Le Guo, and Ming-Ming Cheng. Amt: All-pairs multi-field transforms for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9801–9810, 2023.

[51] Qiuheng Wang, Yukai Shi, Jiarong Ou, Rui Chen, Ke Lin, Jiahao Wang, Boyuan Jiang, Haotian Yang, Mingwu Zheng, Xin Tao, et al. Koala-36m: A large-scale video dataset improving consistency between fine-grained conditions and video content. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8428–8437, 2025.

[52] Yinan Chen, Jiangning Zhang, Teng Hu, Yuxiang Zeng, Zhucun Xue, Qingdong He, Chengjie Wang, Yong Liu, Xiaobin Hu, and Shuicheng Yan. Ivebench: Modern benchmark suite for instruction-guided video editing assessment. *arXiv preprint arXiv:2510.11647*, 2025.

[53] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, WANG HongFa, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *The Twelfth International Conference on Learning Representations*, 2024.

Appendix

Contents

A	**Identity-Aware Memory Implementation Details**	16
B	**Systematic Inference Acceleration Details**	18
C	**LLM/VLM Prompt Templates**	19
D	**NarraStream-Bench Database Construction**	25
E	**Metric Details in NarraStream-Bench**	26
F	**Baseline Implementation Details**	29
G	**Benchmark Evaluation Robustness Analysis**	29
H	**Human Alignment Study Details**	29
I	**Limitation Analysis**	31
J	**More Qualitative Results and Details**	31
K	**Ethics Statement**	36
L	**Reproducibility Statement**	36
M	**Broader Social Impact**	36
N	**Repeated Main Results**	36
O	**Transfer to LongLive**	36
P	**Text-Only Identity Controls**	37
Q	**Independently Authored Scripts**	37
R	**Complex Narrative Subsets**	37
S	**Entity-Weight Sensitivity**	37

A Identity-Aware Memory Implementation Details

This appendix supplements Sec. 3.1 with the implementation of the identity-aware memory module.

Runtime memory state. IAMFlow maintains three data structures during generation. The global entity registry $\mathcal{R}$ maps each persistent ID to a canonical name, a list of textual aliases observed across prompts, a merged attribute list, and per-prompt instances. The frame archive $\mathcal{F}$ stores selected historical frames with their prompt index, associated entity IDs, entity score, VLM visual score, fused score, and per-layer KV cache. The active identity memory m^{id} is a compact list of archived frame IDs. The selected KV tensors are sorted by temporal order, concatenated block-wise, and written into the model memory bank before denoising.

Algorithm 1 IAMFlow Streaming Generation with Identity-Aware Memory

Require: Causal video generator G_θ, prompt sequence $\{p_t\}_{t=1}^{T}$, local window size W_{loc}
Require: Identity-memory budget B, asynchronous verification worker

Initialize global registry $\mathcal{R} \leftarrow \emptyset$, frame archive $\mathcal{F} \leftarrow \emptyset$, local cache $\mathcal{L} \leftarrow \emptyset$
for prompt index $t = 1, \ldots, T$ **do**
  Extract entities and attributes $\mathcal{E}_t \leftarrow \textsc{LLMParse}(p_t)$
  Assign persistent IDs and update registry $(\mathcal{G}_t, \mathcal{R}) \leftarrow \textsc{MatchOrCreateID}(\mathcal{E}_t, \mathcal{R})$
  $m_t^{\mathrm{id}} \leftarrow \textsc{GreedyIDRetrieve}(\mathcal{F}, \mathcal{G}_t, B)$
  Inject the KV cache of m_t^{id} into the model memory bank
  for chunk index $n = 1, 2, \ldots$ under prompt p_t **do**
    $z_n \leftarrow \textsc{DenoiseChunk}(G_\theta, p_t, \mathcal{L}, m_t^{\mathrm{id}})$
    Submit z_n to the asynchronous worker for visual scoring; request attribute check if $n = 1$
    Append z_n to $\mathcal{L}$ and update the clean-context KV cache
    if the local-window update evicts a chunk **then**
      $c \leftarrow$ the evicted chunk
    else
      $c \leftarrow$ the clean-context cache after the context update
    end if
    Collect worker results needed by c; if first-chunk verification returns, correct $\mathcal{R}$
    Build entity-token weights from active IDs $\mathcal{G}_t$ and registry $\mathcal{R}$
    Score frames in c using $s(f) = (1 - \lambda)\hat{s}_{\mathrm{entity}}(f) + \lambda s_{\mathrm{visual}}(f)$
    Archive the top frame and its entity IDs, fused score, and KV cache into $\mathcal{F}$
  end for
end for

Ablation fallback protocols. For the memory-mechanism ablations in Table 3, all unablated components are kept unchanged so that each row isolates a single design choice. In *w/o LLM planning*, we disable both LLM-based entity extraction and LLM-based global ID matching. The system falls back to a deterministic text heuristic: human-related noun phrases and explicitly named characters are extracted from the segment prompt, normalized by lowercasing and stripping articles, and linked across prompts only by exact normalized surface-form matches. Unmatched entities are assigned new segment-level IDs, so this variant does not perform semantic alias resolution, pronoun resolution, or attribute-based identity matching, while the frame archive, identity-aware scoring, retrieval, VLM verification, and prompt transition remain active. In *w/o VLM verification*, we disable asynchronous VLM scoring and attribute correction; archived frames are ranked only by the normalized entity score $\hat{s}_{\mathrm{entity}}(f)$, and the registry is updated only from prompt-side observations. In *w/o memory bank*, we remove the global registry $\mathcal{R}$, frame archive $\mathcal{F}$, active identity memory m^{id}, and all long-term identity-memory retrieval and KV injection. Generation therefore relies only on the sink frames and sliding local window of the base autoregressive generator under the same prompt schedule and transition policy.

Prompt-level entity processing. At the first chunk of each prompt, Qwen3-4B-Instruct extracts human entities and stable visual attributes from the prompt. For the first prompt, all extracted entities are assigned new IDs. For later prompts, explicit novelty markers such as "another", "new", or "different" trigger allocation of a new ID; otherwise, the LLM compares the current entity descriptor with the existing registry and either reuses a matched ID or allocates a new one. When an ID is reused,

IAMFlow appends the new alias and prompt instance to $\mathcal{R}$ and merges newly observed attributes. This converts prompt-level references such as "young man", "the protagonist", and "he" into a shared identity address.

Frame archival and identity-aware scoring. After each generated chunk, IAMFlow archives one representative latent frame. For early chunks before the local cache rolls, the frame is selected from the clean-context KV cache after the context update; afterwards, it is selected from the chunk leaving the local window. For the active prompt, IAMFlow builds an entity-token weight vector $\omega \in \mathbb{R}^S$ with a fixed heuristic. All raw token weights are initialized to 1.0. For each active entity name and extracted attribute, we locate its lowercase character span in the prompt, map the character-span ratio to token indices, and expand the span by $0.02S$ tokens on both sides. Tokens in these matched spans are assigned weight 2.5. Unmatched tokens in the first 8% and last 8% of the prompt are assigned weights 0.7 and 0.5, respectively, while other unmatched tokens remain at 1.0. If no entity or attribute span is matched, tokens from 10% to 85% of the prompt length receive weight 1.5 and all other tokens remain at 1.0; if no entity is detected, all tokens receive weight 1.0. The normalized weights used below are

$$\tilde{\omega}_u = \frac{\omega_u}{\sum_{v=1}^{S} \omega_v + 10^{-8}}. \tag{7}$$

Given cached text keys K^{text} and candidate visual keys K^{vis}, the implementation aggregates the weighted text-key anchor and frame key as

$$\bar{\mathbf{r}}_{\text{id},h} = \sum_u \tilde{\omega}_u \mathbf{K}^{\text{text}}_{u,h}, \quad \bar{\mathbf{k}}_{f,h} = \frac{1}{n_f} \sum_{v=1}^{n_f} \mathbf{K}^{\text{vis}}_{f,v,h}, \tag{8}$$

and scores each candidate frame by

$$s_{\text{entity}}(f) = \frac{1}{H} \sum_{h=1}^{H} \frac{\langle \bar{\mathbf{r}}_{\text{id},h}, \bar{\mathbf{k}}_{f,h} \rangle}{\sqrt{d}}. \tag{9}$$

We use a representative early transformer layer in implementation. This keeps frame ranking lightweight while exploiting the generator's internal text-conditioned key-space compatibility. This corresponds to the single-layer implementation of Eq. 5, with $\mathcal{L} = \{l^\star\}$ and $\beta_{l^\star} = 1$. The entity score is min-max normalized within the chunk and fused with the VLM score as

$$s(f) = (1 - \lambda)\hat{s}_{\text{entity}}(f) + \lambda s_{\text{visual}}(f), \tag{10}$$

where $\lambda = 0.3$. The highest-scoring frame is stored in $\mathcal{F}$ together with its associated entity IDs and KV cache.

Dynamic retrieval and memory injection. At each prompt switch, IAMFlow retrieves archived frames for the active entity IDs using a greedy set-cover strategy. The dynamic budget is the number of frames needed to cover the required IDs, capped by the maximum identity-memory size. When multiple entities are active and enough archive candidates exist, IAMFlow keeps at least two memory frames to avoid collapsing different identities into a single reference. At each greedy step, the selected frame maximizes the number of still-uncovered IDs, with entity score used as the tie-breaker. If the budget is not exhausted after all IDs are covered, remaining slots are filled by high-scoring archived frames. The selected KV caches are concatenated and injected into the memory bank; the sink memory and local sliding window remain unchanged.

Asynchronous VLM verification. IAMFlow decodes generated chunks with a background VAE worker and submits sampled frames to Qwen3-VL-2B-Instruct for visual verification. The VLM scores three sampled frames from each chunk and returns $s_{\text{visual}}(f) \in [0, 1]$ for the fusion above. For the first chunk of each prompt, the same VLM call also verifies the registry attributes against rendered pixels and writes corrected attribute lists back to $\mathcal{R}$. The background worker preserves causal VAE order, while the main DiT stream continues denoising the next chunk. In practice, the three-chunk eviction lag hides most VAE and VLM latency, so visual verification improves memory quality without adding a blocking step to the common generation path.

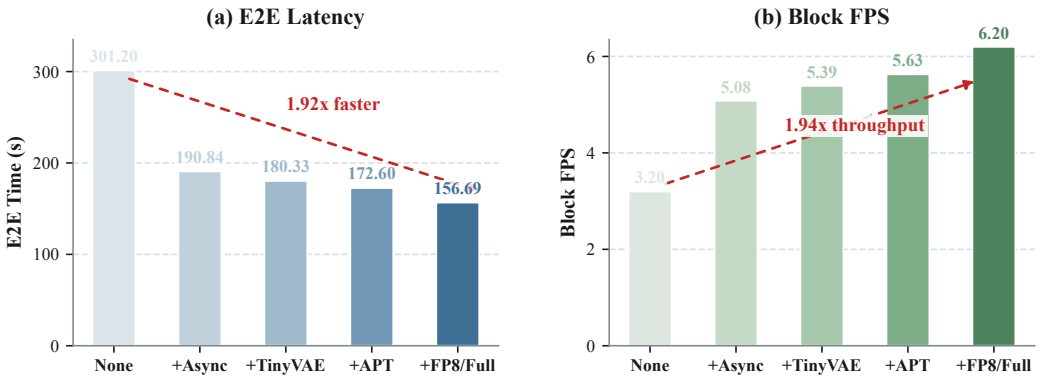

Figure 6: **Build-up analysis of IAMFlow systematic inference acceleration**. Compared with IAMFlow without systematic inference acceleration, the full accelerated IAMFlow substantially reduces end-to-end latency and improves block-level throughput. The build-up starts from the identity-aware memory pipeline without acceleration and progressively adds the Asynchronous Verification Pipeline, TinyVAE decoding, Adaptive Prompt Transition, and FP8 DiT execution.

B Systematic Inference Acceleration Details

This appendix gives the implementation details behind the systematic inference acceleration pipeline summarized in Sec. 3.2. The main design goal is to keep identity-aware memory off the critical generation path: prompt understanding runs only at segment boundaries, visual verification overlaps with DiT denoising, and prompt switches avoid full temporal recaching. Fig. 6 shows that, compared with IAMFlow without systematic inference acceleration, the full accelerated IAMFlow achieves substantial end-to-end speedup and higher block-level throughput.

Adaptive prompt transition. Autoregressive multi-prompt generation usually refreshes the temporal cache at each prompt switch by rerunning the recent latent context through the DiT under the new text condition, so that cross-attention states match the new prompt. This recache pass scales with the local window length and reruns much of the denoising model. IAMFlow replaces it with Adaptive Prompt Transition (APT): before switching prompts, the current cross-attention keys and values are copied into a preallocated buffer; the new prompt cache is lazily initialized on the next forward pass; each cross-attention layer then uses

$$\mathbf{K}_\tau = (1-\alpha_\tau)\mathbf{K}_{\text{old}} + \alpha_\tau \mathbf{K}_{\text{new}}, \quad \mathbf{V}_\tau = (1-\alpha_\tau)\mathbf{V}_{\text{old}} + \alpha_\tau \mathbf{V}_{\text{new}}. \tag{11}$$

This blend operates in the cross-attention conditioning space rather than in pixel or noise space. We use it as a local interpolation between old and new semantic conditioning, so the denoising direction changes continuously after a prompt switch instead of being reset by a hard condition replacement. Let τ denote the number of latent frames generated after a prompt switch and let $d_{\text{delay}} = 3$. During the delay phase, $\alpha_\tau = 0$. Afterwards, α_τ follows a cosine schedule,

$$\alpha_\tau = \frac{1}{2}\left(1 - \cos\left(\pi \frac{\tau - d_{\text{delay}}}{W_{\text{apt}}}\right)\right), \quad \tau \in [d_{\text{delay}}, d_{\text{delay}} + W_{\text{apt}}], \tag{12}$$

and becomes 1 after the transition. The window length is adaptive. Let $\bar{\mathbf{e}}_i$ and $\bar{\mathbf{e}}_{i+1}$ be the mean-pooled text-encoder embeddings of adjacent prompts. We use the clipped cosine distance

$$\delta = \text{clip}\big(1 - \cos(\bar{\mathbf{e}}_i, \bar{\mathbf{e}}_{i+1}), 0, 1\big)$$

and set

$$W_{\text{apt}} = \text{snap}\big(W_{\min} + \delta(W_{\max} - W_{\min})\big), \quad W_{\min} = 3,\ W_{\max} = 15, \tag{13}$$

where $\text{snap}(\cdot)$ aligns the window to the 3-frame chunk boundary. The clipping keeps W_{apt} within the declared window range before snapping. Since prompt embeddings are already computed before generation, this adds no extra text-encoder pass. Small edits therefore use short transitions, while larger scene or role changes receive longer blending windows.

Asynchronous Verification Pipeline. IAMFlow needs pixel-space evidence for VLM visual scoring and for first-chunk attribute correction. If VAE decoding and VLM inference were synchronous, visual verification would directly increase user-facing latency. We instead use a single background worker for streaming VAE decoding and a separate asynchronous VLM executor. After each chunk is denoised, the main DiT stream submits the latent chunk to the background worker and immediately continues to cache update and the next denoising step. The single-worker VAE queue preserves causal decoder order.

The schedule exploits the autoregressive eviction lag. The local portion of the KV cache consists of 3 sink latent frames and a 9-frame sliding local window, so the local window spans three 3-frame chunks; the retrieved identity-memory block is separate from this eviction schedule. For chunk $n \geq 4$, the frame selected for archival comes from the chunk leaving the sliding local window, i.e., chunk $n - 3$. Thus the VAE and VLM have roughly three chunks of DiT computation to finish before their scores are needed. For the first three chunks of a prompt, where no local chunk has been evicted, IAMFlow archives from the clean-context cache after cache update, allowing VAE decoding to overlap with the clean-context forward pass. The VLM request is bounded by sampling three frames from each 12-frame decoded chunk at pixel-frame indices $\{0, 4, 8\}$ and resizing them to 256×256.

Efficient deployment. The efficiency experiments use a LightX2V-style FP8 checkpoint for the DiT linear layers while keeping cache tensors and numerically sensitive operations in BF16/FP32. This targets the dominant DiT matrix-multiplication cost without changing the identity-aware memory paradigm. IAMFlow also separates diffusion and language-model workloads: the DiT, text encoder, and VAE run on the generation device, while the LLM and VLM can be pinned to another device and served through vLLM. The LLM is invoked once per prompt segment for entity extraction and ID matching, and the VLM runs asynchronously on downsampled frames. For output decoding, IAMFlow replaces the default Wan VAE with a TinyVAE adapted to the Wan2.1 latent layout, greatly reducing the memory footprint and processing latency; because chunks are already decoded asynchronously during generation, the final output step only concatenates decoded chunks instead of launching another full-video VAE pass.

Measurement protocol. All efficiency numbers are measured after model loading and warmup. All methods are evaluated under the same two-GPU NVIDIA H20 budget. For baselines without LLM/VLM workloads, the second GPU is used to offload VAE decoding when applicable, so the reported speedups do not come from allocating more hardware to IAMFlow. End-to-end latency is measured from prompt encoding to the complete pixel-level video output, including prompt encoding, LLM entity processing, memory retrieval and injection, DiT denoising, any blocking wait caused by asynchronous VAE/VLM work, and final pixel-video assembly. FPS is computed from chunk-level throughput: in our Wan2.1 setting, each latent chunk corresponds to 12 decoded pixel frames, so FPS is calculated as 12 divided by the average generation time of one chunk. IAMFlow is faster than prior baselines because its systematic inference acceleration pipeline removes full prompt-switch recaching, overlaps visual verification with denoising, reuses chunk-wise decoded pixels for final output, and reduces the dominant DiT compute with FP8 execution.

C LLM/VLM Prompt Templates

This appendix lists the prompt templates used by IAMFlow's language and vision agents, together with the structured prompts used to construct NarraStream-Bench. All runtime agent calls use deterministic decoding. The LLM entity extraction and ID matching calls use temperature 0; the VLM verification call uses temperature 0.0. The implementation parses only JSON objects from the assistant response and discards surrounding Markdown or natural-language explanations.

LLM entity extraction. At the beginning of each prompt segment, IAMFlow sends the current video prompt as the user message and uses the following instruction to extract identity-bearing human entities. We restrict extraction to human characters because the memory bank assigns persistent IDs only to character identities.

LLM Entity Extraction Prompt

```
[System message]
Extract human characters from the video prompt.

RULES:
1. "entities": ONLY human/person characters (man, woman,
   protagonist, etc.)
   - Extract ONLY visual/physical attributes: hair, clothing,
     accessories, body type, age, skin, facial features
   - DO NOT extract behavioral states (walking, nodding, reading,
     sitting) or emotions (quiet, contemplative, happy)
   - Keep entity names short
OUTPUT FORMAT (JSON object only, no explanation):
{"entities": [{"entity": "<name>", "attrs": ["<attr1>", "<attr2>"]}]}

If no humans found, entities should be [].

[User message]
<prompt_text>
```

The expected payload is

$$\{\texttt{entities} : [\{\texttt{entity} : e, \texttt{attrs} : [a_1, \ldots, a_m]\}]\}. \tag{14}$$

The parser first searches for the outer JSON object. If that fails, it accepts the legacy list-only entity format and then falls back to regular-expression extraction of `entity` and `attrs` fields. Malformed entities are dropped rather than propagated into the memory bank.

LLM global ID matching. For the first prompt, IAMFlow allocates new global IDs to all extracted entities. For later prompts, explicit novelty markers such as "another", "other", "new", "different", "second", and "third" allocate a new ID before calling the LLM. All other entities are matched against the current registry with the prompt below.

LLM Global ID Matching Prompt

```
[System message]
Match a new character to existing characters.

TASK: Given a new character description and existing character registry,
determine if they refer to the same person.

MATCHING RULES:
1. Words like "protagonist", "main character", "he", "she" usually
   refer to previously introduced characters
2. Matching clothing or appearance attributes indicates the same person
3. Words like "another", "other", "new", "different" indicate a NEW
   person - return null

OUTPUT FORMAT (JSON only, no explanation):
{"matched_id": <number or null>}

[User message]
New character description:
"<entity>: <attr1>, <attr2>, ..."

Existing characters:
ID <gid>: <alias1>/<alias2>: <attr1>, <attr2>, ...

Does the new character match any existing one? If yes, return the ID.
If no, return null.
Output JSON only: {"matched_id": <number or null>}
```

The implementation accepts a match only when `matched_id` is present and the ID already exists in the registry. Otherwise, IAMFlow allocates a new ID. This conservative fallback prevents an uncertain match from merging two identities and polluting later memory retrieval.

VLM visual scoring and attribute correction. After each generated chunk, IAMFlow samples three frames from the decoded pixels, resizes them to 256×256, and sends them with the prompt below. The sampled frame indices are 0, 4, and 8 in the 12-frame pixel chunk, corresponding to latent frame indices 0, 1, and 2. The VLM returns frame-level visual scores in $[0, 1]$; for the first chunk of a prompt, it also verifies the current global registry attributes.

VLM Frame Scoring and Attribute Verification

```
You are evaluating video frames for visual quality.
The video prompt is: "<prompt_text>"
Key entities: <entity_1>, <entity_2>, ...

Rate each of the <num_frames> frames from 0.0 to 1.0 on how well they
visually represent the entities and match the prompt description.
Consider clarity, consistency, and visual quality.

Also verify these entity attributes against the frames:
  ID <gid> (<name>): <attr1>, <attr2>, ...
If any attribute is incorrect based on what you see, provide corrections.

Respond ONLY with valid JSON:
{"scores": [<float>, ...], "corrections": null}
If there are attribute corrections, use:
{"scores": [<float>, ...],
 "corrections": {"<global_id>": {"corrected_attrs": ["attr1", "attr2"]}}}
```

When the request is not the first chunk of a prompt, the attribute-verification block is omitted and the expected `corrections` value is `null`. The parser clips all scores into $[0, 1]$, pads missing frame scores with 0.5, and ignores malformed correction entries. If VLM inference or parsing fails, IAMFlow uses a neutral score of 0.5 for every sampled frame and applies no registry correction.

NarraStream-Bench prompt generation. For NarraStream-Bench prompt generation, each example is first represented as a structured prompt plan that specifies the scene anchor and setting, character registry, six segment beats, visible and off-screen characters, reference mode, and pose mode. The LLM writes the first segment with a stricter introduction prompt and writes the remaining five segments with a continuity-aware follow-up prompt.

NarraStream-Bench First-Segment Prompt Template

```
You are generating the first segment prompt for a 60-second single-scene
interactive benchmark video.

Requirements:
- Write exactly one English paragraph.
- Output English only.
- Do not use Chinese characters.
- Do not mix English with any other language.
- Keep the final prompt between 20 and 90 English words.
- Prefer 2-3 short sentences.
- Do not exceed 4 short sentences.
- Keep one consistent scene anchor and background stable.
- You MUST copy the exact SCENE_ANCHOR phrase verbatim in the output.
- Do NOT paraphrase or replace the SCENE_ANCHOR wording.
- Do NOT output control labels such as SCENE_ANCHOR:, SCENE_SETTING:,
  REFERENCE_MODE:, BEAT:, or VISIBLE_CHARACTERS:.
- Avoid dialogue and quotation marks.
```

```
- Introduce only the characters that are visible in segment 1.
- Use explicit visual descriptions for visible characters.
- Respect the requested pose mode and do not default to making every
  character sit at a table.
- Keep the wording compatible with LongLive, MemFlow, and IAMFlow:
  restate subject anchors and scene anchors clearly.
- Keep the style close to concise visual narration, not literary prose.
- Output only the prompt text.
```

For follow-up segments, the system message changes the identity policy while keeping the same length and formatting constraints.

NarraStream-Bench Follow-Up Prompt Template

```
You are generating a follow-up segment prompt for a 60-second single-scene
interactive benchmark video.

Requirements:
- Write exactly one English paragraph.
- Output English only.
- Keep the final prompt between 20 and 90 English words.
- Prefer 2-3 short sentences.
- Keep one consistent scene anchor and one consistent venue identity unless
  the beat explicitly specifies a nearby sub-area.
- You MUST copy the exact SCENE_ANCHOR phrase verbatim in the output.
- Do NOT output control labels such as SCENE_ANCHOR:, SCENE_SETTING:,
  REFERENCE_MODE:, BEAT:, or VISIBLE_CHARACTERS:.
- Avoid dialogue and quotation marks.
- Respect the requested reference mode for this segment:
  explicit_description, partial_reference, or ambiguous_reference.
- Do not restate full appearance descriptions for every visible character
  unless identity would become unclear.
- Use at most one brief background reminder sentence.
- Keep visible character identities stable and never invent new appearance
  details for old characters.
- Mention off-screen characters only when needed for entry-exit continuity.
- Respect the requested pose mode and do not collapse every segment into
  the same seated table posture.
- Keep the style concise, visual, and compatible with LongLive, MemFlow,
  and IAMFlow.
- Output only the prompt text.
```

The user message supplies the exact scene anchor, scene setting, main challenge, pose guidance, required reference mode, target beat, visible characters, off-screen characters, the full character registry, and all previous prompts. This makes each segment self-contained enough for text-to-video models while still testing long-range reference resolution.

NarraStream-Bench evaluation prompts. To make the reported NarraStream-Bench metrics reproducible, we list the evaluation prompt templates used by the LLM planner and MLLM/VLM judges. These templates cover segment-importance planning, continuity and occurrence-group extraction, and video-prompt alignment judging. Placeholders such as `{prompt_lines}`, `{prompts}`, and `{num_segments}` are filled by the evaluator before sending the user message.

LLM Segment-Importance Planner Prompt

```
You are the "segment-weight planner" for streaming video evaluation.

The input is a multi-segment prompt sequence for one video. The sequence
usually follows a beginning setup, middle development, and ending closure
```

```
structure. Based only on the prompts, assign each segment an integer
importance score from 1 to 100.

Please focus on:
1. Whether the segment carries a key transition in the before-after
   relationship. If so, assign a higher weight.
2. Whether the segment contains clear, verifiable, and non-substitutable
   action execution, character interaction, object transfer, state change,
   or task progress. If so, assign a higher weight.
3. Whether the segment is more likely to reveal whether the model truly
   understands and follows the prompt. If so, assign a higher weight.
4. Whether the segment is only repetition, setup, or closure. If so,
   assign a lower weight.
5. When semantic conditions are similar, prioritize event development and
   climax stages, respecting narrative structure.
6. The weights should be discriminative.

Requirements:
- The output length must match the number of input segments.
- Each score must be an integer from 1 to 100.
- Do not evaluate video quality; analyze only the prompts themselves.
- Output JSON only. Do not output explanations, Markdown, or any extra text.

Output format:
{"segment_importance":[int,int,int,int,int,int]}

Input prompt sequence:
{prompt_lines}
```

CAC Continuity MLLM Judge Prompt

```
Analyze the following sequence of video segment descriptions and determine
which adjacent segment pairs should preserve visual continuity, such as the
same scene, the same characters, or an obvious continuation.

Input segments:
{prompt_lines}

For each adjacent segment pair, output a Boolean judgment. The length must
be {num_transitions}.

Output format:
{"keep":[true,false,true]}
```

CLC Occurrence-Group MLLM Extractor Prompt

```
Analyze the following sequence of video segment descriptions. Identify the
main entities that appear, and mark the segment indices where each entity
appears (0-indexed).

{prompt_text}

Output JSON only, for example:
{"wizard":[0,1,2,3], "dragon":[1,2,3,4]}
```

VLM Alignment Judge Prompt

```
You are a video evaluation expert. Below is a streaming generated video and
its corresponding prompt sequence.
The system will provide key frames for each video segment in chronological
order. Judge whether each video segment executes the corresponding prompt,
and give an overall score for the entire sequence.

Prompt sequence:
{prompts}

Requirements:
1. ‘segment_scores‘ must be an array of length {num_segments}. The i-th
   element corresponds to the execution score of the i-th prompt segment.
2. All scores must be integers from 1 to 100. 1 means completely
   inconsistent, and 100 means completely consistent.
3. By default, score according to "the key frames of this segment" and
   "the prompt of this segment". Do not add points based on beginning,
   middle, or ending position, narrative structure, or segment importance.
4. If a segment prompt mainly describes continuation, maintenance,
   confirmation, observation, or still being in some state, you may use the
   state established in the previous segment to judge whether the state
   remains valid. Do not mechanically assign a low score only because the
   action magnitude in this segment is small.
5. Scoring must be based on directly observable evidence:
   - Whether the explicit action occurs
   - Whether character interaction occurs
   - Whether object change, object handoff, or state transition occurs
   - Whether the segment prompt is executed at the correct time
   - If the segment is a continuation/maintenance prompt, whether it clearly
     preserves the previously established character relationships, object
     states, and spatial layout without obvious conflicts
6. If only the characters, scene, or general atmosphere match, but the key
   action, interaction, object ownership, handoff receiving relation, or
   state change is unclear, ‘segment_score‘ should not exceed 50.
   For multi-person scenes, as long as character identity, interaction
   target, or object ownership is clearly uncertain, the score should not be
   high even if the overall image looks similar.
7. If there is almost no direct evidence supporting the segment prompt,
   ‘segment_score‘ should be in 1-20.
   If there is only weakly related, generic, or substitutable evidence,
   ‘segment_score‘ should usually be in 20-50, not high.
8. Adjacent segments may have the same score, and large jumps are allowed.
   Do not output an increasing or decreasing score sequence merely to make
   it look smooth. However, do not automatically assign an extremely low
   score to a "continuation holds" segment only because later-segment action
   is weaker.
9. ‘overall_score‘ indicates whether the whole video completes the key
   events in the correct temporal order and maintains later state
   consistency. If the key action, main interaction, object handoff, or
   state transition clearly occurs and later segments do not obviously
   violate the prompt, a medium-high score is allowed even if later segments
   have weaker motion. Conversely, if the video is only generally smooth,
   scene-consistent, and character-similar, but the key event, character
   relationship, or object ownership cannot be confirmed, ‘overall_score‘
   should be clearly reduced.
10. Assign 100 only when character identity, key action, object relation,
    and timing are all very clear and almost unambiguous.

Output format:
{"segment_scores":[int,int,...],"overall_score":int}
```

Planner and judge output schemas. For metric aggregation, the LLM planner sees only the six prompt texts and returns

$$\{\texttt{segment_importance} : [w_1, \ldots, w_6]\}, \quad w_i \in \{1, \ldots, 100\}. \tag{15}$$

The weights are normalized to sum to one. If the returned list has the wrong length, contains non-integer values, or cannot be parsed as JSON, the metric code falls back to uniform or metric-specific mean aggregation.

For VLM alignment in NarraStream-Bench, the judge receives the prompt sequence and the sampled video frames for each segment, then returns

$$\{\texttt{segment_scores} : [q_1, \ldots, q_T], \texttt{overall_score} : q\}, \quad q_i, q \in \{1, \ldots, 100\}. \tag{16}$$

Segment scores measure whether the visible evidence executes the corresponding prompt. The overall score measures whether the whole video completes the key events in order and preserves later state consistency. The parser normalizes these values to $[0, 1]$ and falls back to the legacy scalar-score parser only for older cached responses.

D NarraStream-Bench Database Construction

This appendix describes the construction of the NarraStream-Bench prompt database. The benchmark is designed for narrative streaming generation: a model must keep a stable scene, follow evolving segment-level instructions, preserve entities after disappearance, and resolve partial or ambiguous references.

Scope and dimensions. Each sample is a 60-second single-scene narrative divided into six consecutive 10-second prompt segments. We keep the scene identity fixed across the six segments so that failures are concentrated on streaming memory and instruction following rather than on full scene replacement. The database is text-only: it contains prompt sequences and structured metadata, without scraped videos, personal data, or external visual assets.

Each prompt plan is labeled by six challenge dimensions: *initial character count* (`single`, `double`, `multi`), *interaction complexity* (`static`, `simple_interaction`, `cooperative_task`), *entry-exit dynamics* (`no_entry_exit`, `single_entry_exit`, `multi_entry_exit`), *temporal callback* (`no_callback`, `adjacent_callback`, `long_range_callback`), *character distinguishability* (`high`, `medium`, `low`), and *reference complexity* (`explicit_description`, `partial_reference`, `ambiguous_reference`). The selected combinations keep the benchmark broad enough to cover important marginal cases while avoiding mechanically combined scenarios that are visually unnatural or difficult to judge reliably.

Resources and sampling. The construction starts from three resources: a scene catalog, a blueprint catalog, and character-profile pools. The scene catalog provides 9 stable venues with fixed `scene_anchor` strings, concise background descriptions, and allowed pose modes. The blueprint catalog provides 12 six-segment narrative templates, spanning control cases, common two-person interactions, multi-person cooperative tasks, entry-exit patterns, and long-range callbacks. The profile pools provide names, stable visual attributes, partial references, and ambiguous references for different distinguishability levels.

The official suite contains 324 samples instantiated from 12 scene anchors, 9 blueprints, and 3 variants per scene-blueprint pair, for a total of 1,944 segment prompts. Variants rotate character profiles and pose modes. The pose mode (`seated`, `standing`, or `mixed`) is only a generation-diversity control, not an evaluation dimension.

Prompt plan construction. For every sampled scene-blueprint pair, we first render a structured prompt plan before asking an LLM to write natural-language prompts. The plan records the scene, blueprint, six dimension labels, pose mode, segment beats, and a complete `character_registry`. The registry assigns each character a stable ID, role, display name, appearance description, explicit reference, partial reference, ambiguous reference, and first visible segment. This registry provides the ground truth used by entity-grounding and cross-segment consistency metrics.

The plan also records three alignment structures. The `presence_matrix` specifies which character IDs should be visible in each segment, and `visible_characters_by_segment` exposes this information to the prompt writer. The `callback_edges` field marks temporal dependencies, including adjacent callbacks and long-range callbacks. The `reference_mode_by_segment` field controls whether each segment uses explicit descriptions, partial references, or ambiguous references.

Prompt realization. The final six prompts are generated from the prompt plan using the templates in Appendix C. The first segment must introduce only the characters visible in segment 1 and use explicit visual descriptions. Follow-up segments receive the previous prompt chain, current beat, visible and off-screen character lists, character registry, required reference mode, and pose guidance. This setup keeps each segment self-contained enough for generation while still testing cross-segment memory.

Quality control and repair. After generation, each six-segment chain is validated by an automatic QC pass. The validator checks the segment count, scene-anchor preservation, control-label leakage, non-English text, quotation marks or dialogue, prompt length, sentence count, unregistered character names, first-segment character introduction, and duplicate segment text. Samples that fail QC are regenerated by resampling variant seeds under the same scene-blueprint pair, then rechecked. The final NarraStream-Bench prompt suite contains 324 QC-passed samples and zero failed samples in the released metadata.

E Metric Details in NarraStream-Bench

This appendix specifies the eleven NarraStream-Bench metrics. Each test sample contains a prompt sequence $\{p_i\}_{i=1}^{T}$ and generated video segments $\{V_i\}_{i=1}^{T}$, where $T = 6$ in our benchmark. All metrics lie in $[0, 1]$, with higher values indicating better performance; tables in the main paper multiply them by 100. Tunable hyperparameters appear as symbols, with fixed values stated next to the corresponding formula. Structural normalizations such as division by 255, 2, or 100 remain explicit. The same settings are used for all methods. Lowercase symbols denote segment-, transition-, or entity-level scores, and uppercase S with a metric-specific subscript denotes the final sample-level score.

Narrative-aware Aggregation. NarraStream-Bench weights segment scores by narrative importance. Given the prompt sequence, an LLM planner assigns integer importance values $u_i \in \{1, \ldots, 100\}$, which are normalized as $w_i = u_i / \sum_{j=1}^{T} u_j$. We use $\mathcal{A}_{\text{seg}}$ for segment-level aggregation and $\mathcal{A}_{\text{tr}}$ for transition-level aggregation:

$$\mathcal{A}_{\text{seg}}(\{x_i\}) = \sum_{i=1}^{T} w_i x_i, \qquad \mathcal{A}_{\text{tr}}(\{x_i\}_{i\in\Omega}) = \frac{\sum_{i\in\Omega}(w_i + w_{i+1})x_i}{\sum_{i\in\Omega}(w_i + w_{i+1})},$$

where Ω is the set of valid transitions for the metric. If the planner response is malformed or unavailable, we use the metric-specific fallback aggregation. We use Qwen/Qwen3.5-27B as the LLM planner for segment-importance estimation. The exact planner prompt is listed in Appendix C. Dataset-level metric scores average valid sample scores. Each group score is the macro-average over its metrics, and Overall is the macro-average over the three groups:

$$S_g = \frac{1}{|\mathcal{M}_g|} \sum_{m\in\mathcal{M}_g} S_m, \qquad S_{\text{overall}} = \frac{S_{\text{quality}} + S_{\text{temporal}} + S_{\text{instruction}}}{3},$$

where $\mathcal{M}_g$ is the metric set for group g: Visual Quality uses Subject Consistency (SC), Background Consistency (BC), Temporal Flickering (TF), Motion Smoothness (MS), and Video Temporal Stability Score (VTSS); Temporal Consistency uses Boundary Smoothness (BS), Conditional Adjacent Consistency (CAC), and Conditional Long-range Consistency (CLC); Instruction Compliance uses Entity Grounding (EG), Dynamic Trajectory (DT), and VLM Alignment (VLM).

Subject Consistency (SC) and Background Consistency (BC). SC measures subject appearance stability with DINO features [47], while BC measures scene and background stability with CLIP features [48]. For each metric $m \in \{\text{SC}, \text{BC}\}$, we sample five frames at relative positions

$\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and extract normalized features $\{\mathbf{x}_{i,k}^m\}$. Local consistency ℓ_i^m is the 25th percentile of sampled frame-pair similarities within V_i. The feature medoid among sampled frames gives the representative feature $\mathbf{r}_i^m$. Let $c_{\text{adj}}^m = \rho_{25}(\{\cos(\mathbf{r}_i^m, \mathbf{r}_{i+1}^m)\}_{i=1}^{T-1})$ denote adjacent-transition consistency and $c_{\text{anchor}}^m = \rho_{25}(\{\cos(\mathbf{r}_1^m, \mathbf{r}_i^m)\}_{i=2}^{T})$ denote first-segment-anchor consistency. We fuse them with local consistency as

$$S_m = \alpha_{\text{cons}} \mathcal{A}_{\text{seg}}(\{\ell_i^m\}) + (1 - \alpha_{\text{cons}}) \cdot \frac{c_{\text{adj}}^m + c_{\text{anchor}}^m}{2}. \tag{17}$$

Here ρ_{25} denotes the 25th percentile. We use $\alpha_{\text{cons}} = 0.34$, so the cross-segment weight is 0.66.

Temporal Flickering (TF). TF penalizes frame-to-frame brightness instability after motion compensation. For each segment, we uniformly sample K_{TF} frames. For each adjacent pair (I_t, I_{t+1}), RAFT [49] estimates forward flow $F_{t\to t+1}$ and backward flow $F_{t+1\to t}$. We warp the luminance Y_{t+1} back to frame t and compute

$$e_t(\mathbf{u}) = \frac{|Y_t(\mathbf{u}) - \text{warp}(Y_{t+1}, F_{t\to t+1})(\mathbf{u})|}{255}. \tag{18}$$

Residuals are measured only on in-bound pixels that pass a forward-backward flow consistency check, which avoids treating occlusions or unreliable flow estimates as flickering. We use $K_{\text{TF}} = 8$. If the valid mask covers less than 5% of pixels for a pair, we fall back to the in-bound pixels. The pair error is the ρ_{pair} percentile of valid-pixel residuals. The segment raw error e_i is the ρ_{seg} percentile over pair errors, and the segment score is

$$s_i^{\text{TF}} = \exp(-e_i/\tau_{\text{TF}}). \tag{19}$$

We use $\rho_{\text{pair}} = 90$, $\rho_{\text{seg}} = 84$, and $\tau_{\text{TF}} = 0.5$, where τ_{TF} controls how quickly residual error lowers the score. The final TF score is $S_{\text{TF}} = \mathcal{A}_{\text{seg}}(\{s_i^{\text{TF}}\})$.

Motion Smoothness (MS). MS evaluates whether the motion trajectory is temporally smooth. For each segment, an AMT-S [50] frame interpolation model predicts the midpoint frame between every second frame pair. Let $\hat{I}_{2t+1}$ be the interpolated midpoint and I_{2t+1} the corresponding original frame. The raw interpolation error is

$$e_i = \frac{1}{N_i} \sum_t \text{mean}_{\mathbf{u},c} \, |I_{2t+1}(\mathbf{u}, c) - \hat{I}_{2t+1}(\mathbf{u}, c)|. \tag{20}$$

It is mapped to a normalized segment score by

$$s_i^{\text{MS}} = \exp(-e_i/\tau_{\text{MS}}). \tag{21}$$

We use $\tau_{\text{MS}} = 3.0$. The final MS score is $S_{\text{MS}} = \mathcal{A}_{\text{seg}}(\{s_i^{\text{MS}}\})$.

Video Temporal Stability Score (VTSS). VTSS uses a learned temporal-stability evaluator [51, 52] to produce raw segment scores r_i. We first compute the sample-level raw score $r = \mathcal{A}_{\text{seg}}(\{r_i\})$, then map it to $[0, 1]$ with an anchored linear transform:

$$S_{\text{VTSS}} = \text{clip}\left(\frac{r - r_{\text{low}}}{r_{\text{high}} - r_{\text{low}}}, 0, 1\right). \tag{22}$$

We use $r_{\text{low}} = 0.02$ and $r_{\text{high}} = 0.075$.

Boundary Smoothness (BS). BS measures whether the generated video changes smoothly at prompt boundaries. For the transition between V_i and V_{i+1}, we take the last two frames of V_i and the first two frames of V_{i+1}. RAFT [49] gives the mean flow magnitude before the boundary m_i^-, across the boundary m_i^b, and after the boundary m_i^+. The expected boundary motion is $\bar{m}_i = (m_i^- + m_i^+)/2$, and the normalized transition score is

$$s_i^{\text{BS}} = \exp\left(-\frac{|m_i^b - \bar{m}_i|}{\epsilon_{\text{BS}} + \lambda_{\text{BS}} \bar{m}_i}\right). \tag{23}$$

We use $\epsilon_{\text{BS}} = 0.02$ and $\lambda_{\text{BS}} = 0.5$. The final BS score is $S_{\text{BS}} = \mathcal{A}_{\text{tr}}(\{s_i^{\text{BS}}\}_{i=1}^{T-1})$.

Conditional Adjacent Consistency (CAC). CAC evaluates adjacent segments only when the prompts imply that visual continuity should be preserved. An MLLM judge reads $\{p_i\}$ and returns a Boolean keep flag for each adjacent prompt pair. For each selected segment, we reuse the sampled DINO [47] frame features and define $\mathbf{v}_i$ as their feature medoid, yielding a single representative segment vector. The raw cosine similarity is $c_i = \cos(\mathbf{v}_i, \mathbf{v}_{i+1})$, which is remapped by

$$s_i^{\mathrm{CAC}} = \mathrm{clip}\left(\frac{c_i - \delta_{\mathrm{CAC}}}{1 - \delta_{\mathrm{CAC}}}, 0, 1\right)^{\kappa_{\mathrm{CAC}}}. \tag{24}$$

We use $\delta_{\mathrm{CAC}} = 0.79$ and $\kappa_{\mathrm{CAC}} = 2.0$. The final CAC score is $S_{\mathrm{CAC}} = \mathcal{A}_{\mathrm{tr}}(\{s_i^{\mathrm{CAC}}\}_{i \in \Omega_{\mathrm{CAC}}})$, where Ω_{CAC} contains the selected continuity transitions. We use Qwen/Qwen3-VL-30B-A3B-Instruct as the MLLM judge for selecting continuity-preserving transitions. The exact judge prompt is listed in Appendix C.

Conditional Long-range Consistency (CLC). CLC measures whether an entity remains recognizable when it disappears and later reappears. An MLLM extracts entity occurrence groups $\mathcal{G}_e = \{i_1, \ldots, i_m\}$ from the prompt sequence. For each entity with at least two occurrences, the first occurrence i_1 is the anchor. For each later occurrence i_j, we compute its DINO [47] similarity $c_{e,j}$ to the anchor and remap it as

$$s_{e,j}^{\mathrm{CLC}} = \mathrm{clip}\left(\frac{c_{e,j} - \delta_{\mathrm{CLC}}}{1 - \delta_{\mathrm{CLC}}}, 0, 1\right)^{\kappa_{\mathrm{CLC}}}. \tag{25}$$

We use $\delta_{\mathrm{CLC}} = 0.58$ and $\kappa_{\mathrm{CLC}} = 1.5$. To aggregate reappearances of the same entity, we normalize the planner weights over target occurrences $\{i_2, \ldots, i_m\}$ and lightly shrink them toward uniform weights. We use $\lambda_{\mathrm{CLC}} = 0.7$ for the planner-weighted component and 0.3 for the uniform component, and denote the resulting target-occurrence weight by $\tilde{w}_{e,j}$. Let $\mathcal{E}^+$ be the valid reappearing entities. The entity and sample scores are

$$a_e = \sum_{j=2}^{m} \tilde{w}_{e,j} s_{e,j}^{\mathrm{CLC}}, \qquad S_{\mathrm{CLC}} = \frac{1}{|\mathcal{E}^+|} \sum_{e \in \mathcal{E}^+} a_e. \tag{26}$$

Samples with no valid reappearance group are omitted from the CLC dataset average. We use Qwen/Qwen3-VL-30B-A3B-Instruct as the MLLM judge for extracting long-range entity occurrence groups. The exact extractor prompt is listed in Appendix C.

Entity Grounding (EG). EG checks whether prompt-specified entities and their visible attributes are grounded in the generated segment. For each prompt p_i, an LLM extracts up to four visually verifiable entities and up to five visible attributes per entity. A VLM receives five sampled frames from V_i and returns, for each entity e, a presence score $a_{i,e}$ and an attribute-match score $m_{i,e}$ in $[0, 1]$. The entity score is $g_{i,e} = a_{i,e} m_{i,e}$, and the segment score is

$$s_i^{\mathrm{EG}} = \begin{cases} \frac{1}{|\mathcal{E}_i|} \sum_{e \in \mathcal{E}_i} g_{i,e}, & |\mathcal{E}_i| > 0, \\ \nu_{\mathrm{EG}}, & |\mathcal{E}_i| = 0. \end{cases} \tag{27}$$

We use $\nu_{\mathrm{EG}} = 0.5$. The final EG score is $S_{\mathrm{EG}} = \mathcal{A}_{\mathrm{seg}}(\{s_i^{\mathrm{EG}}\})$. We use Qwen/Qwen3.5-27B for prompt-side entity extraction and Qwen/Qwen3-VL-30B-A3B-Instruct for frame-based visual scoring.

Dynamic Trajectory (DT). DT rewards visual change when the prompt changes substantially and visual stability when the prompt is mostly continuous. LanguageBind [53] encodes each prompt and video segment. For transition i, let $d_i^p = 1 - \cos(\mathbf{t}_i, \mathbf{t}_{i+1})$ and $d_i^v = 1 - \cos(\mathbf{z}_i, \mathbf{z}_{i+1})$, where $\mathbf{t}$ and $\mathbf{z}$ are normalized text and video features. A soft gate determines whether the transition should favor response or stability:

$$\gamma_i = \sigma\left(\frac{d_i^p - \mu_{\mathrm{DT}}}{\tau_{\mathrm{gate}}}\right). \tag{28}$$

Here $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic sigmoid. We use $\mu_{\mathrm{DT}} = 0.25$ and $\tau_{\mathrm{gate}} = 0.05$. The response term is $r_i = 1 - \exp(-d_i^v/\tau_{\mathrm{on}})$, and the stability term is $h_i = \exp(-d_i^v/\tau_{\mathrm{off}})$. We use $\tau_{\mathrm{on}} = 0.02$ and $\tau_{\mathrm{off}} = 0.06$. The transition score is

$$s_i^{\mathrm{DT}} = \gamma_i r_i + (1 - \gamma_i) h_i. \tag{29}$$

The final DT score is $S_{\mathrm{DT}} = \mathcal{A}_{\mathrm{tr}}(\{s_i^{\mathrm{DT}}\}_{i=1}^{T-1})$.

VLM Alignment (VLM). VLM Alignment directly evaluates prompt-following with a multimodal judge. The judge receives the complete prompt sequence and five uniformly sampled frames from each segment, then returns per-segment integer scores q_i and an overall integer score q, all in $\{1, \ldots, 100\}$. After normalization to $[0, 1]$, the segment component is $Q_{\text{seg}} = \mathcal{A}_{\text{seg}}(\{q_i/100\})$. The final VLM score combines segment-level prompt execution and whole-video order/state consistency:

$$S_{\text{VLM}} = \alpha_{\text{VLM}} Q_{\text{seg}} + (1 - \alpha_{\text{VLM}})(q/100). \tag{30}$$

We use $\alpha_{\text{VLM}} = 0.8$, so the overall-score weight is 0.2. We use Qwen/Qwen3-VL-30B-A3B-Instruct as the multimodal judge for VLM Alignment. The exact judge prompt and output schema are listed in Appendix C.

F Baseline Implementation Details

The official implementations of Self Forcing [19], Rolling Forcing [20], and Deep Forcing [12] do not support interactive multi-prompt generation, where prompts change during an ongoing autoregressive rollout. For NarraStream-Bench, we add a common prompt-switching wrapper to these baselines. Each method follows the same six-segment schedule as IAMFlow, generates a 60-second video at 832×480 resolution, and switches prompts at segment boundaries while preserving the generated history.

We do not switch prompts by replacing only the text-encoder tokens, because the temporal KV cache states would remain conditioned on the previous prompt and weaken the baselines. Following LongLive [8], we instead refresh the active causal context with KV-recache at each prompt switch: after encoding the new prompt, we rerun the DiT on the context window used by each method under the new text condition before continuing generation. We keep each baseline's original long-video mechanism unchanged, including Self Forcing's autoregressive rollout model, Rolling Forcing's rolling context update, and Deep Forcing's deep-sink and participative-compression strategy. For the efficiency comparison, we use the same two-GPU deployment budget for baselines and offload VAE decoding to the second GPU when applicable.

G Benchmark Evaluation Robustness Analysis

Because IAMFlow uses Qwen3-VL-2B-Instruct for internal visual verification and attribute correction, a natural concern is that the main NarraStream-Bench judge, Qwen3-VL-30B-A3B-Instruct, may share family-specific visual or textual preferences with the generation-time verifier. To test whether the evaluation conclusion is sensitive to this same-family evaluator choice, we conduct a cross-judge robustness analysis on LongLive, MemFlow, and IAMFlow. We keep the generated videos, prompt sequences, sampled frames, metric rubrics, and aggregation protocol unchanged, and replace only the MLLM judge with independent non-Qwen multimodal evaluators.

The analysis focuses on the NarraStream-Bench metrics that directly use large-model judgments: transition selection for CAC, entity-occurrence extraction for CLC, frame-based entity grounding for EG, and VLM Alignment. As shown in Table 7, IAMFlow remains the best method on CAC, CLC, and VLM Alignment under both non-Qwen judges, and the same trend is preserved under the original Qwen3-VL-30B-A3B judge. EG is consistently close across judges and does not show a Qwen-specific preference for IAMFlow. These results indicate that the reported improvements, especially on the reviewer-targeted VLM Alignment metric, are not an artifact of using a Qwen-family evaluator. They further support the robustness of NarraStream-Bench as an evaluation benchmark.

H Human Alignment Study Details

We conduct the human alignment study to test whether the automatic metrics in NarraStream-Bench agree with human judgments on model ordering. We sample 30 benchmark cases and compare the same three methods as the main comparison: LongLive, MemFlow, and IAMFlow. Each case contains the six-prompt narrative sequence and three 60-second generated videos. We recruit 30 volunteers, and each volunteer evaluates all cases independently. The annotation interface is shown in Fig. 7.

Table 7: **Robustness of NarraStream-Bench conclusions to the choice of MLLM judge.** We keep the generated videos, metric rubrics, and aggregation protocol fixed, and replace only the judge used by CAC, CLC, EG, and VLM Alignment. GPT-5 and Gemini-3.1-Pro are judges from outside the Qwen family; Qwen3-VL-30B-A3B is the original judge used in the main evaluation. Higher scores indicate better performance.

Method	Judge	CAC ↑	CLC ↑	EG ↑	VLM ↑
LongLive	GPT-5	83.05	90.01	58.98	57.02
	Gemini-3.1-Pro	82.99	89.83	55.42	51.99
	Qwen3-VL-30B-A3B	83.06	89.97	62.40	62.22
MemFlow	GPT-5	83.90	89.81	57.67	55.30
	Gemini-3.1-Pro	83.86	89.69	53.88	49.62
	Qwen3-VL-30B-A3B	83.92	89.71	61.93	60.85
IAMFlow	GPT-5	86.09	91.37	58.54	57.53
	Gemini-3.1-Pro	86.06	91.28	54.92	53.85
	Qwen3-VL-30B-A3B	86.07	91.25	62.03	62.61

Annotation protocol. Participants view the full prompt sequence above the generated videos so that they can judge the full set of NarraStream-Bench metrics. The study is blind: model names are hidden and videos are shown as *Video A*, *Video B*, and *Video C*, with the display order shuffled per sample. Participants evaluate the same 11 dimensions defined in Appendix E, covering video quality, temporal consistency, and instruction compliance. For each sample and metric, they rank the three videos from 1 to 3, where rank 1 indicates the best result. Ties are allowed when videos are visually indistinguishable for the current metric.

Pairwise preference aggregation. We convert ranks into pairwise preferences to match the scoring protocol in the main text. For annotator k, sample i, metric m, and methods a and b with ranks $r_{k,i,m,a}$ and $r_{k,i,m,b}$, the induced pairwise score is

$$s_{k,i,m}(a,b) = \begin{cases} 1, & r_{k,i,m,a} < r_{k,i,m,b}, \\ 0, & r_{k,i,m,a} > r_{k,i,m,b}, \\ 0.5, & r_{k,i,m,a} = r_{k,i,m,b}. \end{cases} \tag{31}$$

We first average pairwise scores across annotators for the same sample, metric, and ordered model pair:

$$\bar{s}_{i,m}(a,b) = \frac{1}{|\mathcal{K}_{i,m}|} \sum_{k \in \mathcal{K}_{i,m}} s_{k,i,m}(a,b), \tag{32}$$

where $\mathcal{K}_{i,m}$ denotes annotators with a valid annotation for metric m on sample i. This produces one human preference value for each sample, metric, and model pair.

Alignment with automatic metrics. Let $A_{i,m}(a)$ be the automatic NarraStream-Bench score of method a on sample i and metric m. For the same ordered pair (a,b), we compute the automatic pairwise margin as

$$\Delta A_{i,m}(a,b) = A_{i,m}(a) - A_{i,m}(b). \tag{33}$$

For each metric, we compute Spearman's rank correlation over all samples and all three unordered model pairs:

$$\rho_m = \mathrm{Spearman}\left(\{\bar{s}_{i,m}(a,b)\}_{i,(a,b)}, \{\Delta A_{i,m}(a,b)\}_{i,(a,b)}\right), \tag{34}$$

where i ranges over the 30 selected samples and (a,b) denotes one fixed ordering of each of the three unordered model pairs drawn from $\mathcal{M} = \{\text{LongLive}, \text{MemFlow}, \text{IAMFlow}\}$. Thus each metric is evaluated with 90 pairwise comparison points. Higher ρ_m means that the automatic metric gives pairwise preferences more similar to human annotators. Table 6 reports the resulting correlation for each metric.

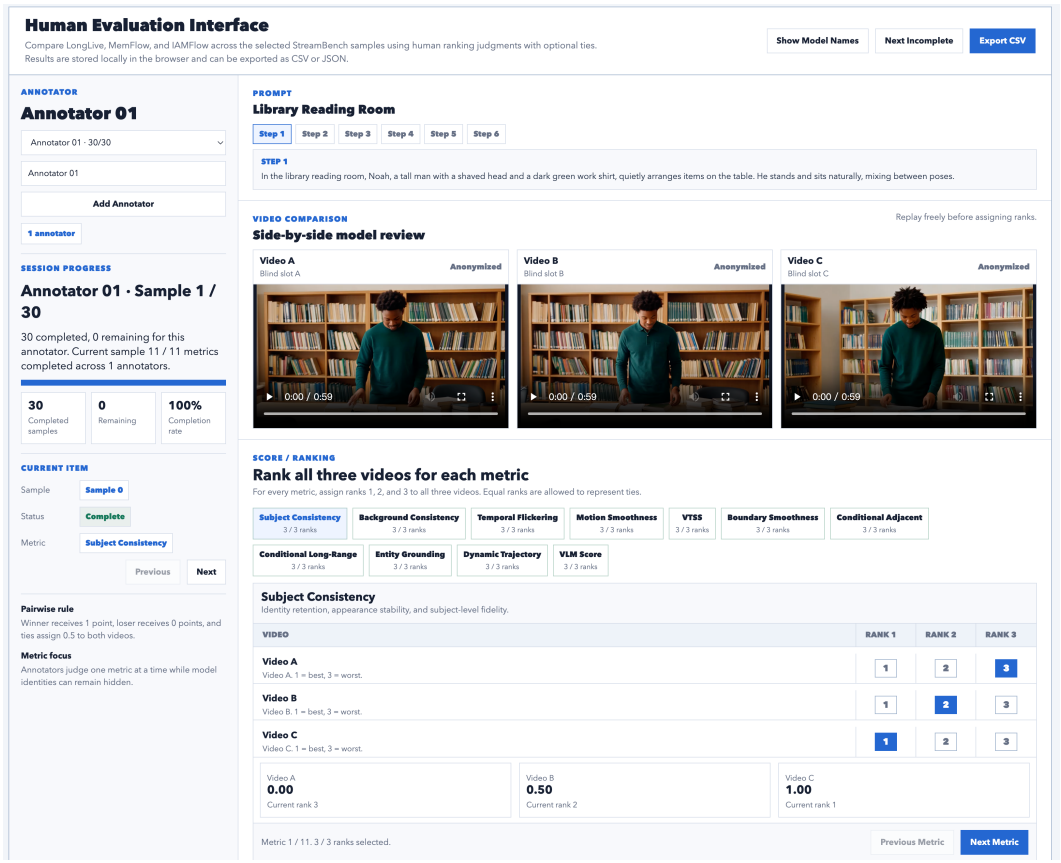

Figure 7: **Human alignment study interface**. Participants view the full prompt sequence and compare anonymized videos before ranking each metric.

I Limitation Analysis

We inspect challenging blueprints with low distinguishability, ambiguous references, and rapidly evolving instructions. Although our method improves long-range consistency through entity- and attribute-centric memory, this design can become overly conservative when the prompt requires fast adaptation to dynamic scene changes or semantically complex edits. In particular, when an entity suddenly changes roles, appearance, or interactions, the retrieved memory may still over-emphasize previously verified identity cues, causing the generator to lag behind the newest instruction. As shown in Fig. 8, IAMFlow often preserves the coarse identity of the main subject, but may miss fine-grained semantic updates such as transient state changes, relational constraints, or newly introduced compositional actions. This tension reflects a trade-off between stable identity preservation and flexible semantic control. Balancing persistent entity memory, prompt responsiveness, and cross-segment coherence remains an open challenge for streaming long-video generation.

J More Qualitative Results and Details

Additional qualitative results of our method are shown in Fig. 9, Fig. 10, and Fig. 11. These examples show that the training-free IAMFlow framework improves long-video identity preservation, temporal consistency, and instruction compliance, producing results on par with or better than training-based baselines.



Figure 8: **Failure case of IAMFlow**. Our method preserves entity identities, but struggles with complex instructions and dynamic actions.

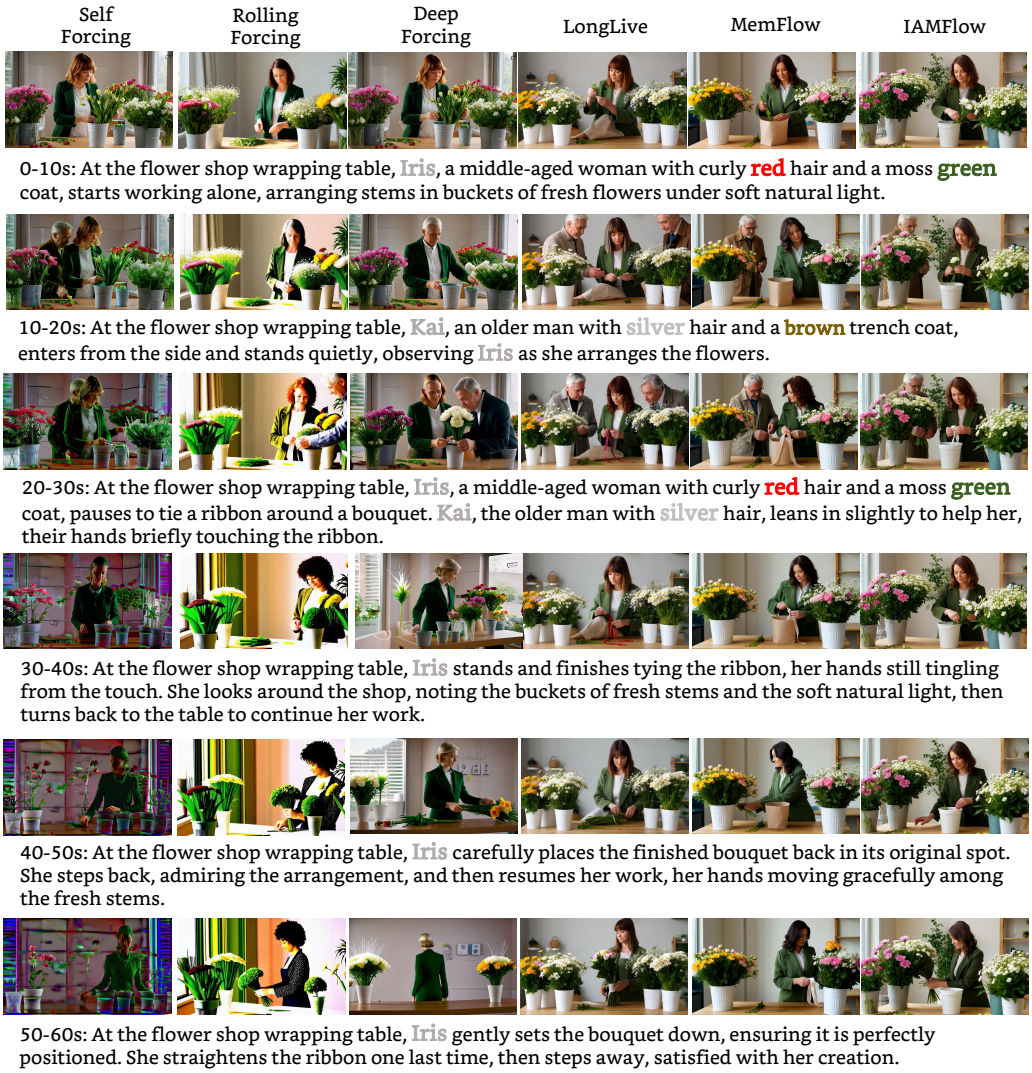

Figure 9: **Two-person interaction scenarios.** IAMFlow preserves the number of entities and their visual attributes across long videos, showing reliable identity-aware memory under evolving two-person interactions.

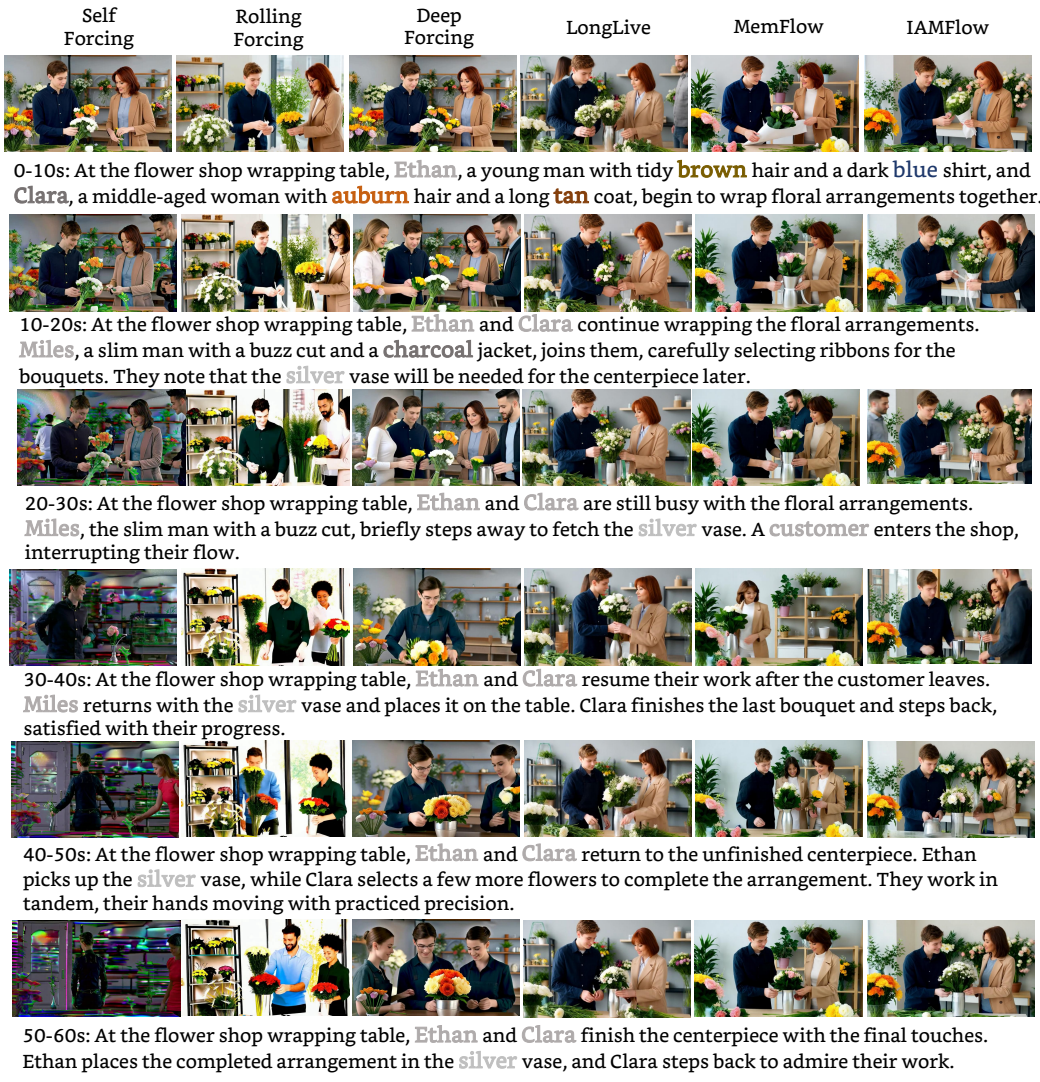

Figure 10: **Multi-person interaction scenarios.** IAMFlow maintains strong long-term consistency and instruction following while keeping multiple interacting entities stable throughout the generation.

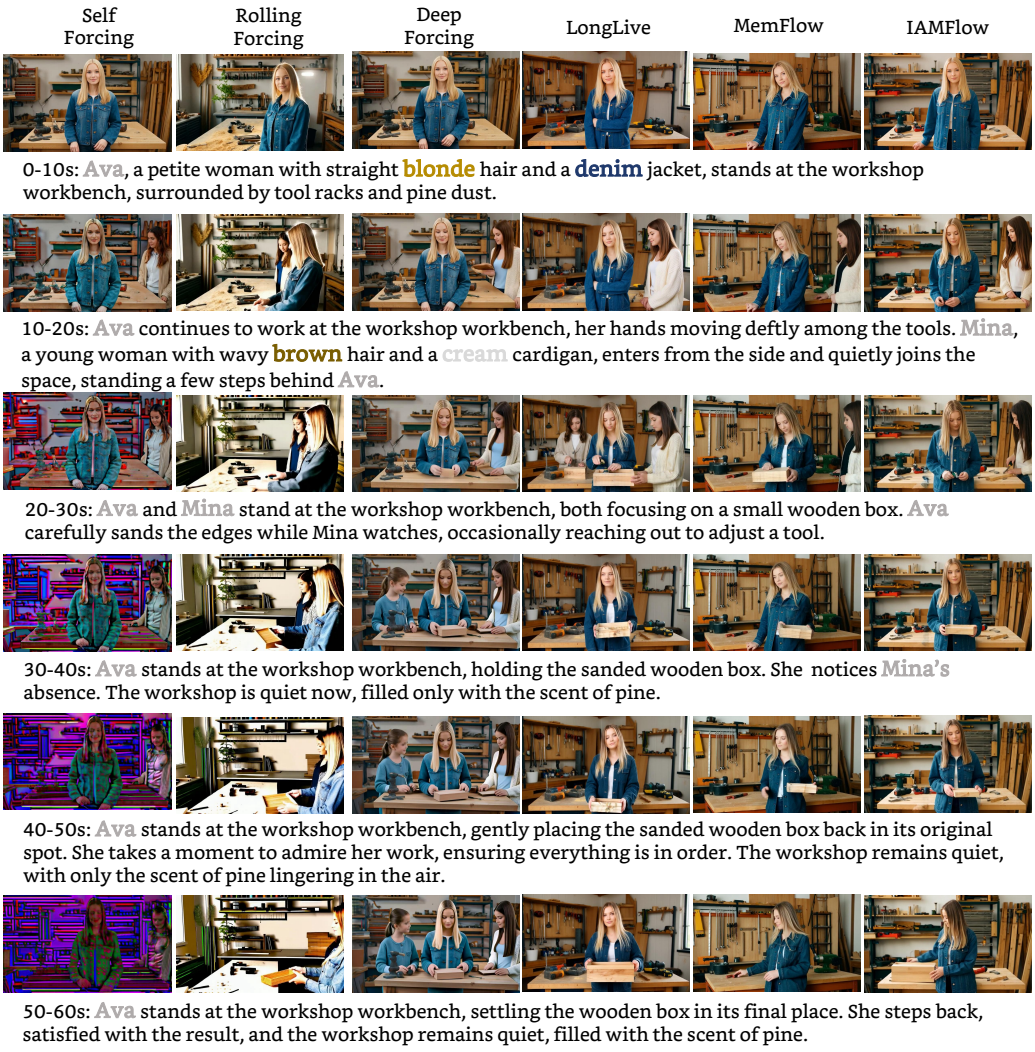

Figure 11: **Two-person interactions across diverse scenes.** IAMFlow generalizes to different scene settings while preserving stable entity identities, attributes, and visual composition across long videos.

K Ethics Statement

IAMFlow is a training-free inference framework and does not introduce additional external video datasets. NarraStream-Bench consists of text-only narrative prompts generated with Qwen2.5-7B-Instruct and Qwen2.5-72B-Instruct. The prompts are intended for academic evaluation and contain no external video data.

L Reproducibility Statement

To support reproducibility, the IAMFlow inference code and NarraStream-Bench evaluation code are publicly available at https://github.com/Eddie0521/IAMFlow and https://github.com/Eddie0521/NarraStream-Bench, respectively. We provide the full implementation details of our method and benchmark in Sec. 3 and Appendices A, D, and E.

M Broader Social Impact

Efficient narrative video generation can lower the cost of long-form visual creation. This may benefit education, storyboarding, accessibility tools, creative prototyping, small studios, and researchers without access to expensive video production pipelines. Since IAMFlow improves long-range consistency without additional video-model training, it may also reduce the compute needed to study interactive long-video generation.

However, the same capabilities can be misused. More coherent long videos may make fabricated events, synthetic identities, or misleading visual narratives harder for viewers to detect. Identity-aware memory can also preserve biased or stereotyped visual attributes if the base generator, LLM, VLM, or benchmark prompts encode such biases. In deployment settings, these risks should be addressed through content provenance, visible labeling, digital watermarking, abuse monitoring, safety filters, and policies that forbid non-consensual likeness generation and deceptive use. We also encourage continued collaboration among researchers, platform operators, policymakers, and civil-society groups on transparent data practices and evaluation protocols for synthetic video.

N Repeated Main Results

To assess variability in generation, we repeated the main comparison using three independent random seeds and matched prompts and sampling settings across methods. Table 8 reports means and standard deviations.

Table 8: Repeated main results reported in the author response.

Method	VQ ↑	TC ↑	IC ↑	Overall ↑
LongLive	82.47 ± 0.41	73.67 ± 0.87	63.74 ± 0.98	73.29 ± 0.74
MemFlow	82.33 ± 0.10	72.34 ± 0.58	65.17 ± 1.41	73.28 ± 0.30
IAMFlow	83.35 ± 0.41	79.91 ± 0.84	63.60 ± 1.50	75.62 ± 0.70

O Transfer to LongLive

To assess transfer across autoregressive backbones, we integrate Identity-Aware Memory into LongLive while keeping the number of retained memory frames constant.

Table 9: Transfer of Identity-Aware Memory to LongLive.

Method	VQ ↑	TC ↑	IC ↑	Overall ↑
LongLive	82.47 ± 0.41	73.67 ± 0.87	63.74 ± 0.98	73.29 ± 0.74
+ Identity-Aware Memory	83.08 ± 0.45	75.77 ± 1.05	64.18 ± 1.20	74.34 ± 0.75

P Text-Only Identity Controls

Identity Reminder appends identity descriptions to the prompt, while LLM Rewrite makes names, coreference relations, and appearance attributes explicit in each segment.

Table 10: Text-only identity controls reported in the author response.

Method	VQ ↑	TC ↑	IC ↑	Overall ↑
MemFlow	82.33 ± 0.10	72.34 ± 0.58	65.17 ± 1.41	73.28 ± 0.30
+ Identity Reminder	81.93 ± 0.41	71.33 ± 0.34	67.07 ± 2.51	73.44 ± 0.98
+ LLM Rewrite	82.45 ± 0.59	72.33 ± 0.58	63.87 ± 0.98	72.88 ± 0.68
IAMFlow	83.35 ± 0.41	79.91 ± 0.84	63.60 ± 1.50	75.62 ± 0.70

Q Independently Authored Scripts

We evaluate on 30 independently authored six-segment scripts written by volunteers unfamiliar with the paper. The volunteers were asked to write coherent six-prompt narratives and were free to choose characters, references, attribute changes, and scene transitions.

Table 11: Results on independently authored scripts.

Method	VQ ↑	TC ↑	IC ↑	Overall ↑
LongLive	78.58	58.68	77.20	71.49
MemFlow	79.20	61.67	76.06	72.31
IAMFlow	79.55	68.49	76.40	74.81

R Complex Narrative Subsets

We further report the challenging NarraStream-Bench subsets involving complex multi-character interactions and long-range entity recall.

Table 12: Results on complex narrative subsets.

Method	VQ ↑	TC ↑	IC ↑	Overall ↑
LongLive	81.90	73.43	62.49	72.61
MemFlow	81.82	72.86	62.76	72.48
IAMFlow	82.74	80.17	63.75	75.55

S Entity-Weight Sensitivity

We report the sensitivity of results to the matched-entity weight.

Table 13: Sensitivity to the matched-entity weight.

Setting	Weight	VQ ↑	TC ↑	IC ↑	Overall ↑
Uniform	1.0	83.29 ± 0.43	79.97 ± 1.22	62.40 ± 1.83	75.22 ± 1.01
Moderate	2.0	83.20 ± 0.50	79.95 ± 0.98	63.38 ± 2.46	75.51 ± 0.83
Default	2.5	83.35 ± 0.41	79.91 ± 0.84	63.60 ± 1.50	75.62 ± 0.70
Strong	3.0	83.20 ± 0.37	79.90 ± 1.40	63.55 ± 2.72	75.55 ± 1.43

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the scope of IAMFlow, NarraStream-Bench, and the reported experimental improvements. The claims are supported by the method description in Sec. 3 and the experimental evidence in the Experiments section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main paper points to challenging failure cases, and Appendix I discusses limitations involving fast semantic changes, complex actions, and the trade-off between identity preservation and prompt responsiveness.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not involve theoretical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sec. 3, the Experiments section, and Appendices A, D, E, and F describe the method, benchmark construction, metrics, baselines, and implementation settings needed to reproduce the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: IAMFlow code and usage instructions are available at https://github.com/Eddie0521/IAMFlow; NarraStream-Bench code and evaluation instructions are available at https://github.com/Eddie0521/NarraStream-Bench.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Experiments section reports the generation setting, resolution, memory sizes, model choices, hyperparameters, baselines, and evaluation protocol. Additional implementation, prompt, benchmark, and metric details are provided in Appendices A–E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix N reports means and standard deviations over three independent random seeds for the repeated main comparison; additional results are reported in Appendices O, P, and S.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the type of compute resources used for all experiments in Sec. 4 and Appendix B to facilitate reproducibility and resource estimation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms to the NeurIPS Code of Ethics. See Appendix K.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix M.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper does not introduce a newly trained high-risk generative model or a scraped dataset; NarraStream-Bench is a text-only prompt benchmark. We discuss them in Appendix M.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original papers of all datasets, models, and code used in this work are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce IAMFlow and NarraStream-Bench, described in the main paper and appendices; their code repositories and usage instructions are publicly available as noted above.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Section 4 and Appendix H describe the human-alignment protocol, participant count, blind display setup, annotation dimensions, tie handling, and score aggregation. The study uses generated videos and benchmark prompts for pairwise perceptual comparison.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [Yes]

Justification: Our study involved preference annotation of AI-generated videos. Participants were informed about the nature of the task, no personal or sensitive information was collected, and the task posed minimal risk to participants. Under our institution's policy, this type of annotation work does not require IRB or equivalent approval.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Sec. 3.1 describes LLM-based entity extraction and ID matching and VLM-based visual verification as core components of IAMFlow. Appendix C provides the prompt templates, decoding settings, output schemas, and parser behavior.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.