STREAMINDEX: DECOUPLING INDEXING FROM ENCODING FOR SCALABLE STREAMING VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Serving concurrent video streams under a fixed GPU budget requires sustained ingestion and timely responses to unpredictable queries. Existing streaming video MLLMs typically encode incoming frames before deciding which information to retain or retrieve, making continuous visual computation a scalability bottleneck. We propose STREAMINDEX, a training-free framework that decouples stream-time indexing from query-time visual encoding. During ingestion, lightweight CPU-side statistics group similar frames into clusters, and only one representative per finalized cluster is encoded for retrieval. When a query arrives, the system retrieves relevant clusters and recovers details through motion-based frame selection and selective reuse of stable spatial features. This design reduces continuous encoding while controlling the additional work on the query path. Experiments on the `Qwen-VL` and `LLaVA-OneVision` families show improved multi-stream scalability with competitive accuracy on OVO-Bench and StreamingBench. In a 16-stream 1080p query-load test at two queries per minute per stream, STREAMINDEX achieves a P95 response latency of 5.57 seconds, while existing state-of-the-art baselines either violate the real-time bound or run out of memory. The results expose a practical trade-off between ingestion cost, query-time computation, and answer quality. The code is available at `https://anonymous.4open.science/r/StreamIndex-7D6E`.

1 INTRODUCTION

While multimodal large language models (MLLMs) (Li et al., 2025; Bai et al., 2025; Wang et al., 2025a; Zhang et al., 2025a) have advanced offline video understanding (Fu et al., 2025a; Wu et al., 2024; Meng et al., 2026), deploying them on continuous video streams (Tong et al., 2026) introduces a different constraint: many streams must share a fixed GPU budget while responding to queries that are unknown during ingestion. Applications such as surveillance, interactive live broadcasting (Yan et al., 2026), and real-time video commentary (Xu et al., 2025; Chen et al., 2025a; Rao et al., 2025) require both sustained stream processing and timely answers. As concurrency, resolution, and video history increase, computation spent before a query arrives can determine whether the system remains usable at all.

Most streaming methods address this setting through query-agnostic visual memory (Zhoubian et al., 2026). *Memory-as-Context* methods (Zhang et al., 2026a; 2025b; Wang et al., 2025b) maintain compact encoded history for direct LLM input, whereas *Retrieval-from-Memory* methods (Di et al., 2025; Ning et al., 2025; Zheng et al., 2025; Zeng et al., 2025) retain encoded history and retrieve relevant evidence after a query arrives. Despite their different memory interfaces, both largely follow an *encode-first, select-later* pipeline: expensive visual encoding precedes the decision about which evidence to retain or use. Figure 1 illustrates the resulting multi-stream bottleneck. Continuous encoding and memory maintenance can make compact-memory systems such as HERMES fall behind live input, while accumulating encoded history can exhaust the memory available to systems such as REKV. This motivates moving the selection decision before full visual encoding.

We propose STREAMINDEX, an *index-first, encode-on-demand* framework that decouples stream-time indexing from query-time visual encoding. The first challenge is to organize the stream without

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

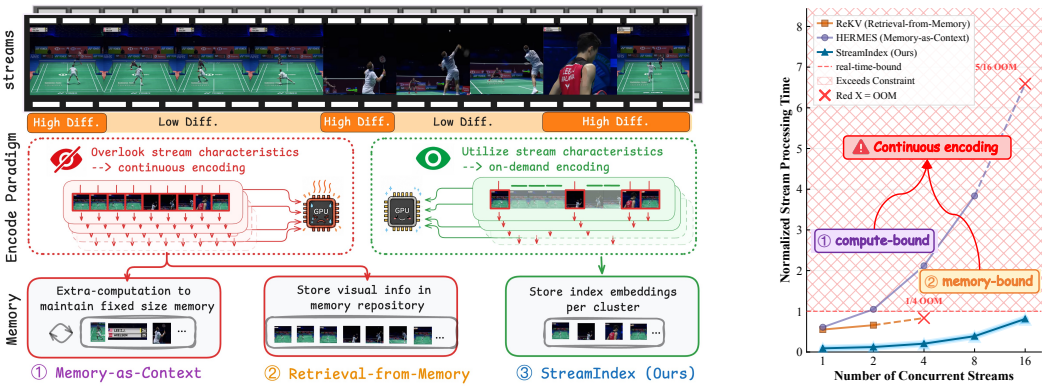


Figure 1: Left: Encode-first memory designs compared with the index-first design of STREAMINDEX. Right: Under increasing concurrency, the compared baselines either exceed the real-time bound or exhaust GPU memory.

encoding every frame. Our Stream Path uses lightweight CPU-side appearance statistics to group similar frames and encodes only one representative per finalized cluster, yielding a coarse index for later retrieval. The second challenge is to recover details without simply shifting excessive computation into the user's waiting time. Our Query Path retrieves relevant clusters and uses a shared motion signal to select a small frame subset and reuse stable spatial features, reserving fresh encoding for motion-intensive regions. Recent frames and summaries of skipped intervals supplement the retrieved evidence when assembling the LLM context.

This design changes where computation is spent: it reduces continuous ingestion cost while accepting on-demand work when queries arrive. We therefore evaluate stream scalability, response latency under increasing query load, and benchmark accuracy separately. The results show improved robustness under demanding multi-stream workloads and competitive response quality across two backbone families, while also exposing the latency advantage of pre-encoded memory in light workloads and the finite serving capacity of our method. Our contributions are:

- We identify continuous stream-time visual encoding as a scalability bottleneck when multiple video streams share a GPU, motivating selection before expensive encoding (Section 4.2).
- We introduce STREAMINDEX, combining CPU-side cluster indexing with query-time retrieval and motion-aware selective reuse to control detail-recovery cost without additional training (Section 3).
- We evaluate the resulting accuracy–efficiency trade-off through multi-stream stress tests, query-load measurements, OVO-Bench and StreamingBench accuracy, and component ablations (Section 4).

2 RELATED WORK

Offline and Streaming Video Understanding. Offline video methods address context limits through iterative evidence search (Yin et al., 2025; Feng et al., 2025; Fei et al., 2025; Zhang et al., 2025c) or query-conditioned memory (Jin et al., 2025; Wang et al., 2025c; Yu et al., 2025). Their direct use in streaming is constrained by the latency of multi-step reasoning or the need to know the query before constructing memory. Streaming systems instead maintain query-agnostic history through compact visual embeddings (Zhang et al., 2025b), KV-cache memories (Kim et al., 2025; Yang et al., 2025; Zhang et al., 2026a; Di et al., 2025; Ning et al., 2025; Chen et al., 2025b), or hierarchical retrieval structures (Jeong et al., 2025; Xue et al., 2025; Zeng et al., 2025; Zheng et al., 2025; Xiong et al., 2025). These designs improve how encoded history is stored and accessed, but generally require visual encoding before memory selection or compression. STREAMINDEX complements this perspective by changing the timing of visual computation: it builds a coarse index during ingestion and recovers selected details after retrieval.

Efficient Vision Transformer (ViT) Encoding. Visual acceleration methods (Zhang et al., 2026b) exploit temporal redundancy through keyframe selection (Hu et al., 2025) or token dropping and

compression (Bagrov et al., 2025; Yao et al., 2025; Shao et al., 2026), and spatio-temporal redundancy through joint token merging or pruning (Huang et al., 2025; Shao et al., 2025; Fu et al., 2025b; Hyun et al., 2025; Khan et al., 2026). These methods use full or shallow-layer visual features to guide reduction, so feature extraction remains part of the selection process. In STREAMINDEX, stream segmentation and query-time frame and token selection are driven by CPU-side appearance and motion statistics. Representative encoding and query-guided retrieval still use learned embeddings, but the redundancy decisions do not require encoding every candidate frame first.

3 METHOD

STREAMINDEX separates coarse evidence indexing from detailed visual processing (Figure 2). The **Stream Path** groups incoming frames using CPU-side statistics and caches one encoded representative per finalized cluster. The **Query Path** retrieves clusters from this index and recovers selected visual details within explicit frame and token budgets.

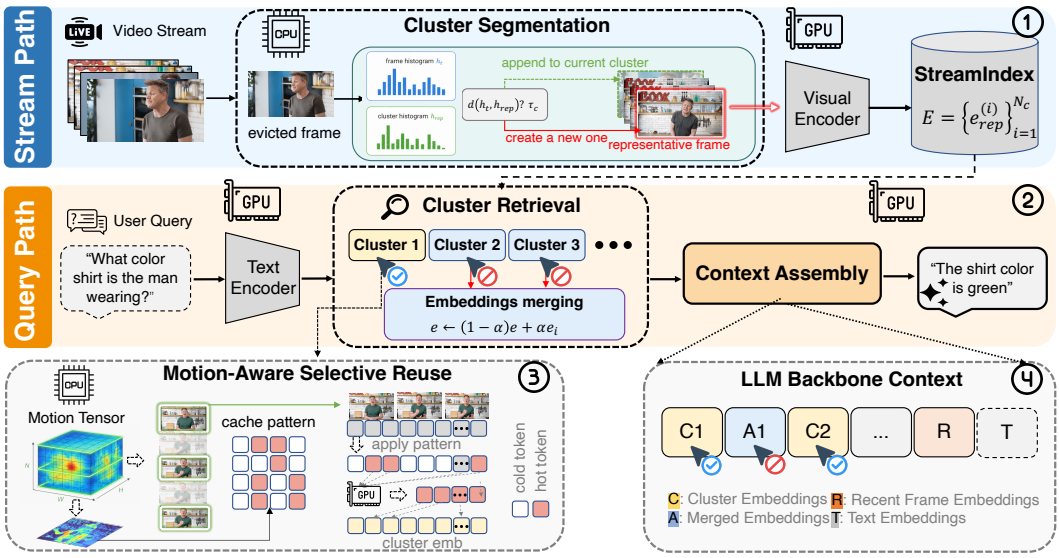


Figure 2: Method overview of STREAMINDEX. ① The Stream Path performs stream-time indexing, whereas the ② Query Path executes on-demand encoding and assembles visual evidence for response generation. ③ Motion-Aware Selective Reuse reduces query-time computation by selecting motion-stable frames and recomputing only hot tokens. ④ The final LLM context assembles selected cluster embeddings, merged gap embeddings, and recent-frame buffer embeddings in temporal order.

3.1 STREAM PATH: LIGHTWEIGHT STREAM INDEX CONSTRUCTION

As illustrated in Figure 2 ①, to avoid encoding every incoming frame, the Stream Path builds a lightweight stream index by clustering similar frames and encoding only one representative frame.

Specifically, the Stream Path maintains a first-in-first-out (FIFO) buffer of size m. When the buffer reaches capacity, the oldest frame f_t is evicted and compared with the active cluster. If the frame remains similar to the active representative and the cluster has not reached its size limit, it is appended to the current cluster; otherwise, the active cluster is closed and a new cluster is started from f_t. The first frame of each new cluster is encoded once by the GPU as its representative. This produces a multimodal embedding $\mathbf{e}_{\text{rep}}^{(i)} \in \mathbb{R}^{H \times W \times D}$, where $H \times W$ denotes the spatial token grid and D is the hidden dimension. As streaming progresses, the representative embeddings of all closed clusters are cached and accumulated into a global stream index $\mathbf{E} = \{\mathbf{e}_{\text{rep}}^{(i)}\}_{i=1}^{N_c}$, where N_c is the number of observed clusters.

The frame-to-cluster comparison uses a compact appearance statistic. Specifically, we extract a 256-bin normalized grayscale histogram $\mathbf{h}_t$ for the evicted frame and measure its distance to the

representative histogram $\mathbf{h}_{\text{rep}}$ using the Bhattacharyya distance:

$$d(\mathbf{h}_{\text{rep}}, \mathbf{h}_t) = -\log \sum_{b=1}^{256} \sqrt{\mathbf{h}_{\text{rep}}(b)\, \mathbf{h}_t(b)}, \tag{1}$$

where histograms are normalized by pixel count; zero overlap gives infinite distance. We compare against the fixed first-frame histogram and append a frame only if $d < \tau_c = 0.1$ and the cluster has fewer than 16 frames; otherwise, we start a new cluster. Appendix B.1 compares this statistic with raw pixel differences and GPU embedding distance, and discusses its limitations.

3.2 Query Path: Retrieval and On-Demand Encoding

When a query arrives, we close the active cluster and add it to the index. Query tokens $\mathbf{Q}$ come from the backbone's input embedding layer; each representative is scored by averaging each query token's maximum cosine similarity to its visual tokens (Appendix B.2). We retrieve the top-k clusters and control detail-recovery cost through frame selection and spatial token reuse.

3.2.1 Motion-Aware Selective Reuse

Temporal frame selection reduces visual encoding and LLM context length; spatial token reuse further reduces fresh visual computation within the selected frames. Both decisions use projections of the same CPU-side motion tensor.

Spatio-Temporal Motion Tensor Construction. As illustrated in Figure 2 ③, given a retrieved cluster $\{f_1, \ldots, f_T\}$, we construct a motion tensor on the $H \times W$ token grid by stacking Gaussian-smoothed frame differences. Let $\phi(\cdot)$ denote grayscale conversion followed by downsampling to the patch level, and $\mathcal{G}_\sigma(\cdot)$ denote Gaussian smoothing. The motion tensor is

$$\mathbf{M} = \operatorname*{stack}_{\tau=1}^{T-1} \mathcal{G}_\sigma\Big(\big|\phi(f_{\tau+1}) - \phi(f_\tau)\big|\Big) \in \mathbb{R}^{(T-1)\times H\times W}. \tag{2}$$

This tensor serves as the common basis for both temporal frame selection and spatial token reuse, allowing the two decisions to be driven by a consistent motion signal rather than separate heuristics.

Temporal Projection for Budget-Driven Frame Selection. To estimate temporal stability, we project $\mathbf{M}$ onto the temporal axis by averaging each motion slice over the spatial dimensions:

$$\mathbf{s}_\tau = \frac{1}{H \times W} \sum_{x=1}^{H} \sum_{y=1}^{W} \mathbf{M}_{\tau,x,y}, \quad \tau = 1, \ldots, T-1, \tag{3}$$

where $\mathbf{s} \in \mathbb{R}^{T-1}$ is the motion score array, and each element s_τ measures the average motion intensity between frame f_τ and $f_{\tau+1}$. Instead of using a fixed motion threshold, we adopt a budget-driven selection strategy. For each retrieved cluster, we retain the top-K_f frames with the lowest motion scores in $\mathbf{s}$. If the cluster length is smaller than the target budget, we simply keep all frames.

Within a retrieved cluster, low-motion frames can provide stable visual states, whereas high-motion frames may capture transient camera or object motion (Figure 3). This is a task-dependent choice: under our evaluated frame budget, low-motion selection gives higher overall accuracy than random or high-motion selection (Table 5).

Spatial Projection for Budget-Driven Token Reuse. After obtaining the selected frame subset $\mathcal{S}_c$, we reuse the same tensor $\mathbf{M}$ to estimate spatial redundancy. Specifically, we aggregate the motion slices associated with the selected frames by mean-pooling along the temporal dimension. Let $\mathcal{I}_c = \{t-1 \mid f_t \in \mathcal{S}_c,\ t > 1\}$ denote the corresponding motion-slice indices. This produces a 2D spatial motion map $\mathbf{P}_c \in \mathbb{R}^{H\times W}$:

$$\mathbf{P}_c[x, y] = \text{mean}_{\tau\in\mathcal{I}_c}\, \mathbf{M}_{\tau,x,y}, \tag{4}$$

where (x, y) indexes spatial token locations. We then construct a binary mask $\mathbf{B} \in \{0, 1\}^{H\times W}$, using a token reuse ratio $K_p \in (0, 1]$: the top $\lceil (1 - K_p)HW \rceil$ most motion-intensive positions in

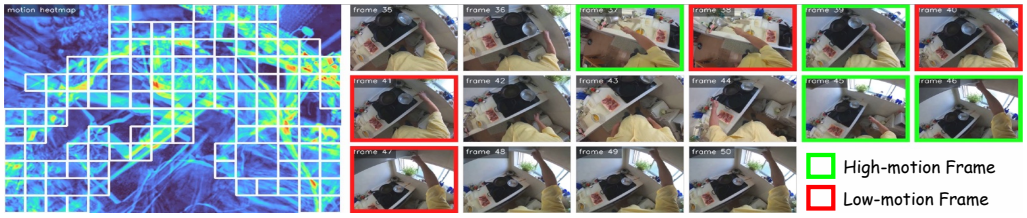


Figure 3: Left: Spatial projection of motion tensor visualization, where the white grids indicate hot tokens. Right: Comparison of frame selection strategies.

$\mathbf{P}_c$ are marked as *hot* tokens for recomputation, and the remaining are treated as *cold* tokens. For a selected frame f_t, let $\mathbf{F}_{t,x,y}$ denote the newly computed feature at a hot position, and let $\mathbf{e}^{(c)}_{\text{rep}}$ denote the cached representative feature map. The final reused feature map is defined element-wise as

$$\tilde{\mathbf{F}}_{t,x,y} = \begin{cases} \mathbf{F}_{t,x,y}, & \mathbf{B}_{x,y} = 1 \quad \text{(hot)} \\ \mathbf{e}^{(c)}_{\text{rep},x,y}, & \mathbf{B}_{x,y} = 0 \quad \text{(cold)} \end{cases}. \tag{5}$$

Hot patches are selected before visual encoding, and their outputs are scattered into the cached feature map. This approximates full-frame encoding; backbone-specific patch, position, and attention handling is detailed in Appendix B.3.

The frame budget K_f and reuse ratio K_p jointly control the cost and fidelity of detail recovery. Section 4.5 evaluates these mechanisms, and Appendix E studies their parameter sensitivity.

CPU Work. Stream-time comparisons use fixed-size histograms; query-time motion processing is linear in cluster length for a fixed token grid. These operations avoid per-frame GPU feature extraction for selection, although their practical cost also depends on preprocessing, input resolution, and concurrency. The query-load evaluation includes CPU preprocessing and queueing in the measured response latency.

3.2.2 Context Assembly and Response Generation

Retrieved clusters may be separated by unselected intervals. For each such gap, we form a compact embedding $\mathbf{A}_j$ by applying an exponential moving average (EMA) to its cached representatives in chronological order, with update rate α. This provides coarse context for skipped footage without encoding additional frames; the recurrence is specified in Appendix B.4.

Let $\mathbf{C}_{s_i}$ contain the final features of selected cluster s_i, $\mathbf{R}$ the fine-grained embeddings of recent buffered frames, and $\mathbf{T}$ the query embedding. We interleave the selected clusters and gap summaries in temporal order and append the recent context and query:

$$\mathcal{X} = [\mathbf{A}_0, \mathbf{C}_{s_1}, \mathbf{A}_1, \ldots, \mathbf{C}_{s_k}, \mathbf{A}_k, \mathbf{R}, \mathbf{T}], \tag{6}$$

omitting $\mathbf{A}_j$ when no gap exists.

4 Experiments

To our knowledge, no established benchmark jointly measures accuracy and latency under varying independent stream counts and query rates on a fixed GPU budget. We therefore test answer quality on OVO-Bench and StreamingBench, and serving scalability under controlled loads. With identical inputs and inference settings, no frame loss, and no truncation, additional load affects waiting time, not the model's conditional answer distribution or expected accuracy.

4.1 Experimental Setup

We evaluate training-free methods with `Qwen2.5-VL-7B` and `LLaVA-OV-7B` as the primary backbones, comparing StreamIndex with ReKV, STC, and HERMES. Experiments use a single NVIDIA A100 80GB GPU; streaming methods ingest video at 1 FPS unless otherwise stated. Answer

quality is measured on the Real-time, Backward, and Forward tasks of OVO-Bench and the Real-time task of StreamingBench. The default configuration uses $k = 2$ retrieved clusters, at most $K_f = 4$ selected frames per cluster, token reuse ratio $K_p = 0.8$, and recent-frame buffer size $m = 4$.

We distinguish stream processing from answering a query. For the resolution stress test, normalized stream-processing time above one means that ingestion falls behind the input video's real-time bound. *Query-to-response latency*, also called user-perceived latency, includes retrieval, on-demand visual encoding, LLM generation, and waiting for queueing or stream-processing backlog, ending when the complete answer is available. Detailed configurations and protocols are provided in Appendix A; limitations and broader impacts are discussed in Appendix F.

4.2 Scalability Across Streams, Resolutions, and Histories

Multi-Stream Deployment Under Different Resolutions. We first test whether coarse indexing allows more concurrent streams to keep pace with incoming video. The two backbones expose different sources of pressure: `Qwen2.5-VL-7B` uses dynamic resolution, while `LLaVA-OV-7B` pools visual embeddings to a fixed token count per frame. We resize a three-minute 1440p video to 480p, 720p, 1080p, and 1440p, increase stream concurrency, and measure stream-processing time normalized by video duration. Query requests across streams are submitted simultaneously in this stress test (Figure 4 and Table 1).

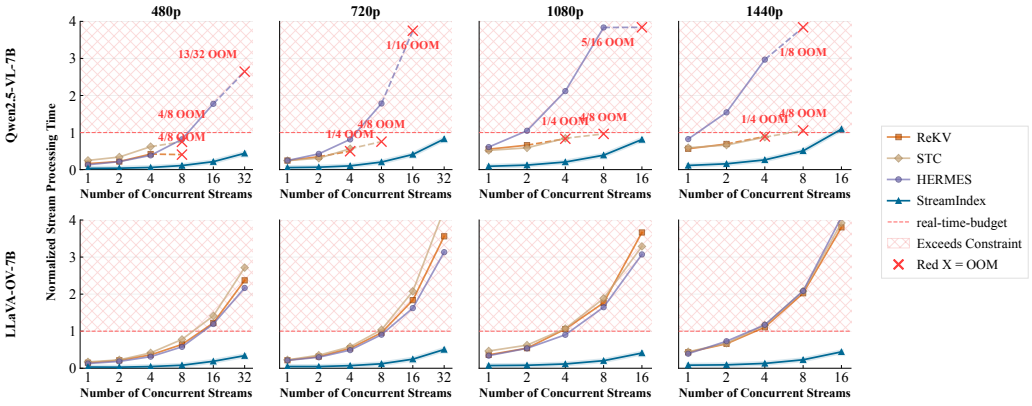


Figure 4: Multi-stream scalability under varying resolutions on `Qwen2.5-VL-7B` and `LLaVA-OV-7B`. Query requests across streams are submitted in parallel at the same timestamp; the plots measure normalized stream-processing time.

Figure 4 shows two failure modes of encode-first processing: frequent OOM failures on `Qwen2.5-VL-7B` as resolution and concurrency rise, and growing processing lag on `LLaVA-OV-7B` even when fixed-token pooling avoids OOM. StreamIndex reduces this ingestion pressure and remains within the real-time bound across the tested settings on both backbones.

The response latencies in Table 1 expose the corresponding trade-off. Under light workloads, pre-encoded memory can answer faster: for example, HERMES outperforms StreamIndex at 480p with four streams on both backbones. As workloads become heavier, stream-processing backlog or OOM can outweigh this query-time advantage. StreamIndex satisfies the real-time bound in all listed settings, although its on-demand encoding still incurs a response cost. Thus, the gain is greater deployment capacity rather than uniformly lower latency at every operating point.

Multi-Stream Deployment Across Video Lengths. We next examine the impact of video length. This experiment evaluates whether methods that appear viable on short streams remain responsive as the stream history grows. We use `Qwen2.5-VL-7B` and measure user-perceived latency on 1-, 3-, 5-, and 10-minute videos under four resolution-concurrency settings.

Figure 5 shows that stream length exposes the cumulative cost of continuous encoding. ReKV quickly runs into OOM failures once the stream becomes longer or concurrency increases. STC remains feasible for several short or moderate settings but also fails under 8-stream deployment.

Table 1: Multi-stream deployment performance across resolutions. Each entry reports end-to-end query-to-response (user-perceived) latency a and response-generation latency b in seconds as $a(b)$; red values violate the real-time bound. Succ. counts non-OOM settings that satisfy the real-time bound out of eight backbone–resolution combinations (two backbones and four resolutions) in each subtable.

(a) Stream = 4

RES.	REKV	STC	HERMES	STREAMINDEX
	Qwen2.5-VL-7B			
480p	1.24 (1.24)	0.91 (0.91)	**0.31** (0.31)	2.75 (2.75)
720p	OOM	0.89 (0.89)	**0.37** (0.37)	6.10 (6.10)
1080p	OOM	**0.67** (0.67)	206 (0.36)	15.2 (15.2)
1440p	OOM	**0.44** (0.44)	364 (0.40)	21.4 (21.4)
	LLaVA-OV-7B			
480p	0.54 (0.54)	0.22 (0.22)	**0.07** (0.07)	3.52 (3.52)
720p	0.51 (0.51)	0.20 (0.20)	**0.07** (0.07)	3.68 (3.68)
1080p	17.0 (6.37)	13.5 (0.22)	**3.19** (3.19)	4.68 (4.68)
1440p	25.6 (5.80)	30.1 (0.20)	36.9 (5.28)	**4.67** (4.67)
Succ.	3/8	6/8	5/8	**8/8**

(b) Stream = 8

RES.	REKV	STC	HERMES	STREAMINDEX
	Qwen2.5-VL-7B			
480p	OOM	OOM	**1.09** (1.09)	6.27 (6.27)
720p	OOM	OOM	146 (0.95)	**13.0** (13.0)
1080p	OOM	OOM	525 (0.79)	**37.2** (37.2)
1440p	OOM	OOM	OOM	**31.5** (31.5)
	LLaVA-OV-7B			
480p	0.53 (0.53)	0.23 (0.23)	**0.07** (0.07)	2.99 (2.99)
720p	0.54 (0.54)	7.14 (0.21)	**0.06** (0.06)	2.38 (2.38)
1080p	148 (6.51)	164 (0.26)	123 (3.06)	**3.84** (3.84)
1440p	195 (5.45)	200 (0.22)	206 (5.33)	**4.84** (4.84)
Succ.	2/8	1/8	3/8	**8/8**

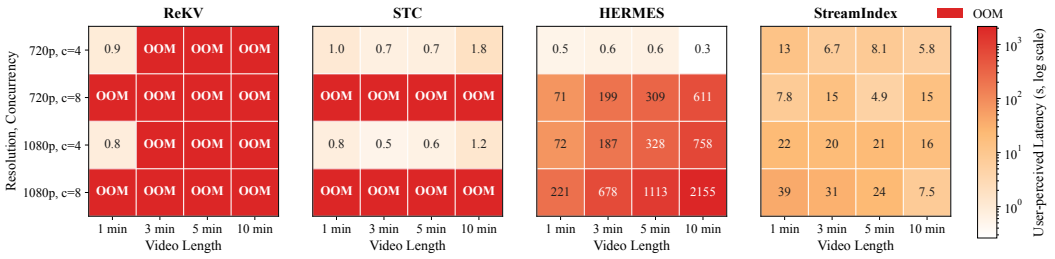


Figure 5: Heatmap of user-perceived latency. Results are shown in seconds (log scale) across various video lengths, resolutions, and concurrency levels on Qwen2.5-VL-7B.

HERMES avoids some OOM failures, yet its user-perceived latency grows to hundreds or thousands of seconds because stream processing falls behind the live input. STREAMINDEX avoids OOM across all tested length settings and keeps user-perceived latency within tens of seconds, indicating better scalability with both concurrency and accumulated video history. In Appendix D, we further evaluate STREAMINDEX on a heterogeneous workload with varied resolutions and durations.

4.3 RESPONSE LATENCY UNDER QUERY LOAD

Reducing ingestion cost is useful only if deferred encoding does not create an excessive query backlog. We therefore compare HERMES, REKV, and STREAMINDEX while varying both stream count and per-stream query arrival rate (Table 2). Each method uses Qwen2.5-VL-7B, 1080p source video at 1 FPS, eight-core CPU affinity, and a single FIFO worker for indexing and queries on one A100 80GB GPU, retaining its own visual processing and caching strategy. Each load point uses 60 seconds of history followed by 600 seconds of measurement, with one run at seed 0. Stream starts and query arrivals are uniformly staggered across time, with the query rate specified per stream; this differs from the simultaneous requests in the resolution stress test. We report P50/P95 latency from scheduled arrival to full-answer completion; RT fail marks real-time-bound violations and OOM marks out-of-memory failures, consistent with the failure categories in Table 1. Additional serving checks are described in Appendix A.

At two queries per minute per stream, STREAMINDEX has the lowest reported P50 and P95 at every tested stream count. At eight streams and four queries per minute per stream, its P50/P95 is 1.90/4.34 seconds, compared with 20.53/31.14 for REKV and 4.51/7.17 for HERMES. At 16 streams and two queries per minute per stream, STREAMINDEX reports P50/P95 of 2.78/5.57 seconds; HERMES violates the real-time bound and REKV runs out of memory. At 16 streams and four queries per

Table 2: Multi-stream deployment under varying query loads on `Qwen2.5-VL-7B`. Stream starts and query arrivals are uniformly staggered across time; query rates are per stream. Each numeric entry reports P50/P95 end-to-end query-to-response latency in seconds, from scheduled arrival to full-answer completion, including queueing. RT fail marks real-time-bound violations; OOM denotes out-of-memory failures. Both are status labels, not latency values. Bold marks the lowest reported latency for each quantile in each row.

Streams	Queries/min per stream	REKV (P50/P95)	HERMES (P50/P95)	STREAMINDEX (P50/P95)
1	0.5	**1.62/1.76**	**1.62**/2.54	1.71/1.86
1	2	1.71/1.91	1.78/2.16	**1.39/1.74**
1	4	1.84/2.21	2.10/2.63	**1.38/1.80**
4	0.5	1.58/**2.16**	1.77/2.29	**1.56**/2.21
4	2	1.86/2.30	1.99/2.67	**1.29/1.64**
4	4	1.90/2.55	1.89/2.84	**1.34/1.67**
8	0.5	1.64/3.20	**1.54**/3.17	1.55/**3.00**
8	2	2.68/4.22	2.47/3.94	**1.64/2.97**
8	4	20.53/31.14	4.51/7.17	**1.90/4.34**
16	0.5	OOM	17.14/25.14	**1.89/3.58**
16	2	OOM	RT fail	**2.78/5.57**
16	4	OOM	RT fail	RT fail

Table 3: Answer quality on the two primary backbones. Values are accuracy (%); Overall follows the full benchmark table in Appendix C. Streaming methods use 1 FPS. †: adaptive frame sampling at 1.0/0.5/0.2 FPS for short/medium/long videos (Appendix A). Bold marks the best score within each backbone.

Method	OVO-Bench			StreamingBench	
	Real-time	Backward	Forward	Real-time	Overall
`Qwen2.5-VL-7B`					
Vanilla†	60.70	42.48	41.93	72.44	57.69
REKV	63.65	46.16	39.23	72.84	58.18
STC	60.66	44.38	35.72	71.00	55.97
HERMES	63.06	**46.67**	**42.90**	72.84	59.08
STREAMINDEX	**71.84**	46.41	39.53	**74.08**	**60.38**
`LLaVA-OV-7B`					
Vanilla†	63.76	43.86	50.20	72.04	60.40
REKV	64.60	46.45	47.16	69.52	59.35
STC	60.31	46.62	47.14	64.20	55.92
HERMES	**69.62**	**48.53**	**51.08**	**74.80**	**63.60**
STREAMINDEX	65.27	47.24	50.03	70.60	60.45

minute per stream, STREAMINDEX also violates the real-time bound, exposing its finite serving capacity.

4.4 ANSWER QUALITY

We next evaluate whether coarse indexing and selective recovery retain sufficient evidence to answer questions. Table 3 reports accuracy for the two primary backbones; Appendix C contains the full model comparison and category-level visualization. This evaluation measures answer quality independently of serving load.

On `Qwen2.5-VL-7B`, STREAMINDEX obtains the highest Overall score among the compared training-free methods, improving from 59.08 for HERMES to 60.38, with its strongest results on real-time questions. On `LLaVA-OV-7B`, it remains close to vanilla inference (60.45 vs. 60.40) but

Table 4: Component ablation of STREAMINDEX. Total Encoded Frames represents the overall cost, while Query-to-Response Encoded Frames represents the cost for lazy encoding. Query-to-Response Latency measures end-to-end time as defined in Appendix A.

Recent Frame	Gap Sum.	Frame Sel.	Token Reuse	OVO-Bench Accuracy (%)				Total Enc. Frames	Query-to-Resp. Enc. Frames	Query-to-Resp. Latency (s)
				Real	Bwd.	Fwd.	Total			
✓				69.40	46.11	40.12	51.88	51.63	18.49	1.07
✓	✓			69.98	46.49	39.98	52.15	51.63	18.49	1.10
✓	✓	✓		71.20	48.88	40.06	**53.38**	43.19	10.05	0.77
✓	✓	✓	✓	71.84	46.41	39.53	52.59	38.33	**5.20**	**0.71**

Table 5: Ablation on frame selection and token reuse strategies on `Qwen2.5-VL-7B`.

Frame Selection Strategy	OVO-Bench Acc				Reuse Mask Strategy	OVO-Bench Acc			
	Real.	Bwd.	Fwd.	Total		Real.	Bwd.	Fwd.	Total
Random frame	71.05	46.36	40.33	52.58	Random reuse	70.82	45.51	39.02	51.78
High-motion frame	70.43	47.21	40.42	52.69	Hot token reuse	71.42	46.52	39.14	52.36
Low-motion frame	**71.20**	**48.88**	40.06	**53.38**	Cold token reuse	**71.84**	46.41	39.53	**52.59**

trails HERMES (63.60). The per-task scores also show that its real-time gains do not imply uniform improvements in backward or forward reasoning. The evidence therefore supports competitive answer quality alongside improved scalability, with backbone- and task-dependent accuracy trade-offs.

4.5 ABLATION OF SELECTIVE RECOVERY AND CONTEXT

Component Analysis. Table 4 separates the contributions of auxiliary context and selective recovery. Gap summarization adds a small accuracy gain (51.88% to 52.15%) without reducing encoded-frame counts. Adding motion-based frame selection improves OVO-Bench total accuracy to 53.38%, reduces query-time encoded frames from 18.49 to 10.05, and lowers latency from 1.10 to 0.77 seconds. Token reuse then reduces query-time encoded frames to 5.20 and latency to 0.71 seconds, while accuracy falls to 52.59%. Frame selection improves both quality and efficiency in this experiment; token reuse provides a further computational saving with a measurable accuracy cost.

Frame Selection and Token Reuse Strategy. We study the effect of frame selection and token reuse strategy in Table 5. For frame selection, we compare random, high-motion, and low-motion strategies under the same budget $K_f = 4$ without token reuse. Results show that the low-motion frame strategy achieves the best overall accuracy, which supports our hypothesis that low-motion frames provide more stable and representative visual evidence. We further compare different token reuse policies under low-motion frame selection with the same reuse ratio of 0.8. Cold-token reuse achieves the highest total accuracy, but its gain over hot-token reuse is modest. This suggests that the cold-token reuse strategy is generally beneficial, while its effect can be limited in cases where the representative frame deviates from the selected frames: tokens at the same spatial grid location may no longer correspond to the same semantic region, making direct token reuse less reliable.

We retain $K_p = 0.8$, $K_f = 4$, $m = 4$, and $k = 2$ as the default operating point; Appendix E reports the full sensitivity analysis and the accuracy–latency trade-offs when these budgets change.

5 CONCLUSION

We present STREAMINDEX, a training-free framework that decouples stream-time indexing from query-time visual encoding. CPU-side clustering reduces continuous encoding, while retrieval and motion-aware selective reuse control the cost of recovering visual details. The evaluations show improved multi-stream scalability and competitive benchmark accuracy, with explicit trade-offs in light-load latency, feature fidelity, and maximum sustainable query load. Adapting indexing and recovery budgets to video content and query demand is a promising direction for further work.

AI USE STATEMENT

We used generative AI tools to polish the manuscript for clarity, grammar, and readability, and to assist with the implementation of experimental code. We did not use generative AI tools for any other research tasks, including developing theoretical models or conceptual frameworks, formulating mathematical claims or proofs, proposing or refining hypotheses, designing research methodology or experiments, generating synthetic datasets, translating text, cleaning or reformatting datasets, conducting qualitative or thematic analysis, or interpreting results. The authors reviewed the AI-assisted edits to ensure that they preserved the intended scientific meaning, and reviewed and tested the AI-assisted code to verify its correctness. The authors take full responsibility for the final content of this work, including the AI-assisted text and code, and for the accuracy and integrity of the reported results.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 and Appendix B describe stream-time indexing, retrieval scoring, motion-aware frame selection, sparse visual encoding, and context assembly. Appendix A documents the benchmarks, backbone models and baselines, implementation settings, hardware and inference configurations, and evaluation protocols, including the query-load procedure and its random seed. Appendix D specifies the heterogeneous-stream workload, while Section 4.5 and Appendix E report component ablations and hyperparameter sensitivity analyses. The code is available through the anonymous repository linked in the abstract.

ETHICS STATEMENT

This work studies efficient streaming video understanding using existing public benchmarks and pretrained models. All experiments use existing OVO-Bench and StreamingBench videos to simulate streaming input offline; we do not collect new video data, recruit participants, connect to live cameras, or deploy the system for real-world surveillance. We verified the applicable usage terms for the datasets, model weights, and third-party code used in this study. Our anonymous repository releases method code without redistributing third-party videos, datasets, or model weights.

Reducing the cost of multi-stream video processing may support beneficial applications, but can also facilitate privacy-sensitive surveillance, unauthorized behavioral profiling, and deployments that amplify model or data biases. Model errors and hallucinations further limit the suitability of the system for decisions with serious consequences. Any future deployment should account for privacy, consent, and the limitations of the underlying models. Appendix F discusses these limitations and broader impacts, and Appendix F.3 describes the assets and licenses. The authors report no funding, commercial collaborations, or conflicts of interest related to this work.

REFERENCES

Natan Bagrov, Eugene Khvedchenia, Borys Tymchenko, Shay Aharon, Lior Kadoch, Tomer Keren, Ofri Masad, Yonatan Geifman, Ran Zilberstein, Tuomas Rintamaki, Matthieu Le, and Andrew Tao. Efficient video sampling: Pruning temporally redundant tokens for faster VLM inference. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2510.14624.

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. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2511.21631.

Joya Chen, Ziyun Zeng, Yiqi Lin, Wei Li, Zejun Ma, and Mike Zheng Shou. Livecc: Learning video LLM with streaming speech transcription at scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Chen_LiveCC_Learning_Video_LLM_with_Streaming_Speech_Transcription_at_Scale_CVPR_2025_paper.html.

Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. Streamingtom: Streaming token compression for efficient video understanding. *CoRR*, 2025b. URL https://doi.org/10.48550/arXiv.2510.18269.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video kv-cache retrieval. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025. URL https://openreview.net/forum?id=8g9fs6mdEG.

Hao Fei, Shengqiong Wu, Wei Ji, Hanwang Zhang, Meishan Zhang, Mong-Li Lee, and Wynne Hsu. Video-of-thought: Step-by-step video reasoning from perception to cognition. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2501.03230.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2503.21776.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Fu_Video-MME_The_First-Ever_Comprehensive_Evaluation_Benchmark_of_Multi-modal_LLMs_in_CVPR_2025_paper.html.

Tianyu Fu, Tengxuan Liu, Qinghao Han, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. Framefusion: Combining similarity and importance for video token reduction on large visual language models. *CoRR*, 2025b. URL https://doi.org/10.48550/arXiv.2501.01986.

Kai Hu, Feng Gao, Xiaohan Nie, Peng Zhou, Son Tran, Tal Neiman, Lingyun Wang, Mubarak Shah, Raffay Hamid, Bing Yin, and Trishul Chilimbi. M-LLM based video frame selection for efficient video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Hu_M-LLM_Based_Video_Frame_Selection_for_Efficient_Video_Understanding_CVPR_2025_paper.html.

Xiaohu Huang, Hao Zhou, and Kai Han. Prunevid: Visual token pruning for efficient video large language models. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.1024/.

Jeongseok Hyun, Sukjun Hwang, Su Ho Han, Taeoh Kim, Inwoong Lee, Dongyoon Wee, Joon-Young Lee, Seon Joo Kim, and Minho Shim. Multi-granular spatio-temporal token merging for training-free acceleration of video llms. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2507.07990.

Soyeong Jeong, Kangsan Kim, Jinheon Baek, and Sung Ju Hwang. Videorag: Retrieval-augmented generation over video corpus. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.1096/.

Hongbo Jin, Qingyuan Wang, Wenhao Zhang, Yang Liu, and Sijie Cheng. Videomem: Enhancing ultra-long video understanding via adaptive memory management. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2512.04540.

Abuzar Khan, Ahmad Junaid, Abid Iqbal, Ghassan Husnain, Wonseop Shin, and Sangsoon Lim. A survey of dynamic token computation in transformers: Taxonomy, stability, and budget-aware evaluation. *IEEE Access*, 2026. URL https://doi.org/10.1109/ACCESS.2026.3720106.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. Infinipot-v: Memory-constrained KV cache compression for streaming video understanding. *CoRR*, 2025. doi: 10.48550/ARXIV.2506.15745. URL https://doi.org/10.48550/arXiv.2506.15745.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *Trans. Mach. Learn. Res.*, 2025. URL https://openreview.net/forum?id=zKv8qULV6n.

Jiahao Meng, Yue Tan, Qi Xu, Kuan Gao, Weisong Liu, Yanwei Li, Jason Li, Lingdong Kong, Haochen Wang, Qianyu Zhou, Jiangning Zhang, Guangliang Cheng, Yunhai Tong, Lu Qi, and Minghsuan Yang. Watch, remember, reason: Human-view video understanding with mllms. *CoRR*, 2026. URL https://doi.org/10.48550/arXiv.2606.07433.

Zhenyu Ning, Guangda Liu, Qihao Jin, Wenchao Ding, Minyi Guo, and Jieru Zhao. Livevlm: Efficient online video understanding via streaming-oriented KV cache and retrieval. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2505.15269.

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video llms with active real-time interaction via disentangled perception, decision, and reaction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Qian_Dispider_Enabling_Video_LLMs_with_Active_Real-Time_Interaction_via_Disentangled_CVPR_2025_paper.html.

Jiayuan Rao, Haoning Wu, Hao Jiang, Ya Zhang, Yanfeng Wang, and Weidi Xie. Towards universal soccer video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Rao_Towards_Universal_Soccer_Video_Understanding_CVPR_2025_paper.html.

Kele Shao, Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Holitom: Holistic token merging for fast video large language models. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2505.21334.

Kele Shao, Keda Tao, Kejia Zhang, Sicheng Feng, Mu Cai, Yuzhang Shang, Haoxuan You, Can Qin, Yang Sui, and Huan Wang. A survey of token compression for efficient multimodal large language models. *Trans. Mach. Learn. Res.*, 2026. URL https://openreview.net/forum?id=G2od9JVHkE.

Junlong Tong, Zilong Wang, YuJie Ren, Peiran Yin, Hao Wu, Wei Zhang, and Xiaoyu Shen. From static inference to dynamic interaction: A survey of streaming large language models. In *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, 2026. URL https://doi.org/10.18653/v1/2026.findings-acl.498.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, Min Dou, Kai Chen, Wenhai Wang, Yu Qiao, Yali Wang, and Limin Wang. Internvideo2.5: Empowering video mllms with long and rich context modeling. *CoRR*, 2025a. URL https://doi.org/10.48550/arXiv.2501.12386.

Yiyu Wang, Xuyang Liu, Xiyan Gui, Xinying Lin, Boxue Yang, Chenfei Liao, Tailai Chen, and Linfeng Zhang. Accelerating streaming video large language models via hierarchical token compression. *CoRR*, 2025b. URL https://doi.org/10.48550/arXiv.2512.00891.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for LLM reasoning on long videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 2025c. doi: 10.1109/CVPR52734.2025.00311. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VideoTree_Adaptive_Tree-based_Video_Representation_for_LLM_Reasoning_on_Long_CVPR_2025_paper.html.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/329ad516cf7a6ac306f29882e9c77558-Abstract-Datasets_and_Benchmarks_Track.html.

Haomiao Xiong, Zongxin Yang, Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Jiawen Zhu, and Huchuan Lu. Streaming video understanding and multi-round interaction with memory-enhanced knowledge. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025. URL https://openreview.net/forum?id=JbPb6RieNC.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. Streamingvlm: Real-time understanding for infinite video streams, 2025. URL https://doi.org/10.48550/arXiv.2510.09608.

Zhucun Xue, Jiangning Zhang, Xurong Xie, Yuxuan Cai, Yong Liu, Xiangtai Li, and Dacheng Tao. Adavideorag: Omni-contextual adaptive retrieval-augmented efficient long video understanding. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2506.13589.

Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, Wang Lin, Hao Liao, Xing Xie, Tao Jin, and Jianxun Lian. Proact-vl: A proactive videollm for real-time ai companions, 2026. URL https://arxiv.org/abs/2603.03447.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. Streammem: Query-agnostic KV cache memory for streaming video understanding. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2508.15717.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, Lingpeng Kong, Qi Liu, Yuanxing Zhang, and Xu Sun. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia, MM 2025, Dublin, Ireland, October 27-31, 2025*, 2025. URL https://doi.org/10.1145/3746027.3754839.

Yufei Yin, Qianke Meng, Minghao Chen, Jiajun Ding, Zhenwei Shao, and Zhou Yu. Videoarm: Agentic reasoning over hierarchical memory for long-form video understanding. *CoRR*, 2025. doi: 10.48550/ARXIV.2512.12360. URL https://doi.org/10.48550/arXiv.2512.12360.

Xinlei Yu, Chengming Xu, Guibin Zhang, Zhangquan Chen, Yudong Zhang, Yongbo He, Peng-Tao Jiang, Jiangning Zhang, Xiaobin Hu, and Shuicheng Yan. Vismem: Latent vision memory unlocks potential of vision-language models. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2511.11007.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. Streamforest: Efficient online video understanding with persistent event memory. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9loSPaBwGO.

Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, Peng Jin, Wenqi Zhang, Fan Wang, Lidong Bing, and Deli Zhao. Videollama 3: Frontier multimodal foundation models for image and video understanding. *CoRR*, 2025a. URL https://doi.org/10.48550/arXiv.2501.13106.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-vstream: Efficient real-time understanding for long video streams. *CoRR*, 2025b. URL https://doi.org/10.48550/arXiv.2506.23825.

Haowei Zhang, Shudong Yang, Jinlan Fu, See-Kiong Ng, and Xipeng Qiu. HERMES: KV cache as hierarchical memory for efficient streaming video understanding. *CoRR*, 2026a. URL https://doi.org/10.48550/arXiv.2601.14724.

Jun Zhang, Yicheng Ji, Feiyang Ren, Yihang Li, Bowen Zeng, Zonghao Chen, Ke Chen, Lidan Shou, Gang Chen, and Huan Li. Efficient inference for large vision-language models: Bottlenecks, techniques, and prospects. In *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, 2026b. URL https://doi.org/10.18653/v1/2026.findings-acl.1057.

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. *CoRR*, 2025c. URL https://doi.org/10.48550/arXiv.2505.18079.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Naishan Zheng, Jie Huang, Qingpei Guo, and Feng Zhao. Videoscaffold: Elastic-scale visual hierarchies for streaming video understanding in mllms. *CoRR*, 2025. URL https://doi.org/10.48550/arXiv.2512.22226.

Sining Zhoubian, Dan Zhang, Evgeny Kharlamov, and Jie Tang. Memory for large language models. *CoRR*, 2026. URL https://doi.org/10.48550/arXiv.2607.25380.

A EXPERIMENTAL SETUP

This appendix expands the setup summarized in Section 4.1, including benchmark protocols, implementation choices, and the query-load procedure.

Benchmarks. We evaluate STREAMINDEX on two streaming video understanding benchmarks, OVO-Bench and StreamingBench. For OVO-Bench, we report results on all three official subtasks: Real-time, Backward, and Forward question answering. For StreamingBench, we evaluate the Real-time subtask, which best matches the online query setting considered in this work. Unless otherwise specified, streaming methods process videos at 1 FPS, following common practice in online video understanding. Accuracy is computed using the official evaluation protocol of each benchmark.

Models and Baselines. STREAMINDEX is model-agnostic and can be instantiated with both the `LLaVA-OV` (Li et al., 2025) and `Qwen-VL` (Bai et al., 2025) model families. We evaluate it with the same backbone models used by the compared training-free systems. We consider two groups of baselines. The first group consists of training-based streaming video methods, including DISPIDER (Qian et al., 2025), FLASHVSTREAM (Zhang et al., 2025b), and STREAMFOREST (Zeng et al., 2025), all based on `Qwen2VL-7B`. The second group consists of training-free methods built on top of open-source MLLMs, including vanilla `LLaVA-OneVision` or `Qwen-VL` inference, REKV (Di et al., 2025), STC (Wang et al., 2025b), and HERMES (Zhang et al., 2026a). All baselines are reproduced from their open-source implementations using their released best-performing configurations and evaluated under the same hardware and inference configuration for fair comparison.

Adaptive Frame Sampling for Base Models. For vanilla base-model inference, encoding every frame at 1 FPS can exceed the multimodal context budget as video duration increases. We therefore adapt the sampling rate to video length: 1.0 FPS for short videos, 0.5 FPS for medium videos, and 0.2 FPS for long videos. The † symbol in Tables 3 and 6 denotes this temporal sampling schedule. Streaming methods ingest video at 1 FPS unless otherwise stated.

Implementation Details. For STREAMINDEX, the stream-time clustering module uses a FIFO buffer of size $m = 4$ and a Bhattacharyya-distance threshold $\tau_c = 0.1$. The default cache ratio is set to 0.8. To avoid overly large clusters whose visual content cannot be adequately represented by a single representative frame, we impose a hard constraint on the cluster size: each cluster contains at most 16 frames. At query time, we retrieve the top-2 clusters for each user query and select at most 4 motion-stable frames from each retrieved cluster. The recent-frame memory keeps the same number of frames as the stream buffer. For gap summarization, we use EMA with update coefficient $\alpha = 0.1$. For response generation, we use deterministic decoding with temperature set to 0.00. We implement our method based on SGLang v0.5.9, and the forward inference module of the underlying models uses SGLang's infrastructure. All models are served with SGLang (Zheng et al., 2024) in bfloat16 precision on a single NVIDIA A100 80GB GPU.

Evaluation Protocol and Metrics. We follow the official evaluation protocols of OVO-Bench and StreamingBench to assess response quality. As our main analysis focuses on deployment scalability under streaming workloads, we summarize the key accuracy results in Section 4.4 and provide the full benchmark comparison in Appendix C. For scalability analysis, we use the original video duration as the real-time bound; a method satisfies the streaming constraint only if its total stream-processing time does not exceed the input video length. We report stream-processing time normalized by video duration, out-of-memory failures, and user-perceived latency. Throughout our experiments, *query-to-response latency* is the total wall-clock time from when the user submits a question until the user receives the complete answer. It includes retrieval, on-demand visual encoding, LLM generation, and all waiting time, including queueing and waiting for any stream-processing backlog. This definition applies to the user-perceived latency in the scalability experiments and the query-to-response latency in the component ablation and sensitivity analysis. For component ablations, we additionally use encoded-frame counts as a diagnostic proxy for visual computation.

Query-Load Protocol. The load test in Section 4.3 compares the three completed suites for HERMES, REKV, and STREAMINDEX. Each method runs on one A100 80GB GPU with `Qwen2.5-VL-7B`, 1080p source video ingested at 1 FPS, eight-core CPU affinity, and a single

FIFO worker for indexing and queries, while retaining its own visual processing and caching strategy. Each load point uses 60 seconds of history followed by 600 seconds of measurement, with one run at seed 0. Stream start times and query arrivals are uniformly staggered across time, and each stream receives queries at the rate listed in Table 2. The stream count therefore denotes concurrent video streams, not the number of queries submitted at the same timestamp; the resolution stress test in Figure 4 uses simultaneous query requests across streams. Latency starts at the scheduled request arrival and ends when the full answer is complete, including queueing. In Table 2, RT fail marks settings that violate the real-time bound, while OOM denotes out-of-memory failures; both are status labels rather than measured latency quantiles. These categories align with the real-time violations and OOM failures in Table 1.

The load-test procedure uses a 30-second response budget. Its pass criteria require at least 95% on-time completion, second-half query-backlog and video-lag slopes no greater than 0.005 queries/s and 0.05 s/s, respectively, and satisfaction of request/frame-loss and resource checks. Table 2 reports latency quantiles and marks real-time violations and OOM failures rather than reporting aggregate pass/fail outcomes, so numeric latency entries should not be interpreted as certifying every check. OVO-Bench and StreamingBench provide answer-quality evaluation protocols, but do not define the controlled multi-stream query-load sweep used here. We therefore evaluate accuracy separately in Section 4.4 and use this load test to measure serving latency and capacity. For a fixed query and visual context, unchanged model parameters and decoding settings define the same conditional answer distribution regardless of queue length; additional waiting alone does not reduce inference accuracy. This reasoning assumes that serving preserves the evidence available at the query timestamp and does not drop frames, truncate context, or interrupt generation. Accuracy and evidence coverage are not measured jointly with load in this test. The query-load, resolution, and video-length tests vary different workload factors, and their latency values should be interpreted within their respective protocols.

B Indexing, Retrieval, and Encoding Details

B.1 Appearance Statistic for Cluster Segmentation

The Stream Path uses a 256-bin normalized grayscale histogram and Bhattacharyya distance to decide whether an incoming frame remains similar to the active cluster representative. Figure 6 compares this CPU-side statistic with raw pixel difference and GPU embedding cosine distance computed with `Qwen2.5-VL-7B`. In the illustrated sequence, raw pixel difference exhibits dense local fluctuations, whereas the histogram distance more closely follows the broad changes in embedding distance. This qualitative comparison motivates a low-cost appearance statistic for segmentation; it does not establish equivalence to semantic embeddings across videos. Grayscale histograms discard spatial arrangement and can miss subtle state changes, which is one reason to retain query-time detail recovery and impose a maximum cluster size. The evaluated cluster-size limit is 16 frames; other implementation settings are listed in Appendix A.

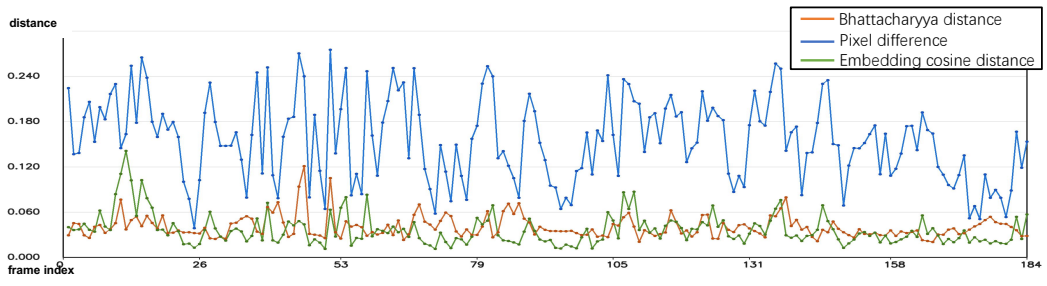


Figure 6: Metric comparison for cluster segmentation. We compare raw pixel difference, Bhattacharyya distance, and GPU embedding cosine distance computed with `Qwen2.5-VL-7B`.

B.2 Query-Time Finalization and Retrieval Scoring

When a query arrives, the active cluster is finalized before retrieval: it is appended to the completed-cluster list and its representative is made available to the retrieval index. The representative is the cluster's first frame and is encoded when the cluster is created, so finalization does not require encoding it again. Frames still in the FIFO buffer remain separate recent context. Thus, frames that have left the buffer but belong to the previously active cluster are eligible for retrieval at the current query.

Retrieval Features. The query is tokenized with the backbone's tokenizer, and its token IDs are mapped through the frozen language-model input embedding layer to obtain $\mathbf{Q} \in \mathbb{R}^{S \times D}$. This step uses embedding lookup rather than a contextual language-model forward pass. For `Qwen2.5-VL`, representative visual tokens are taken after the visual encoder's merger; for `LLaVA-OV`, they are taken after multimodal projection and spatial pooling. The `Qwen3-VL` adapter uses the last returned DeepStack feature for retrieval, while retaining the main and DeepStack features for generation. The adapter matches the retrieval feature dimension to the text embedding dimension when necessary using fixed pooling or zero-padding, with no learned retrieval projection. We distinguish these retrieval features from the features assembled for LLM generation, even when the two coincide for a backbone.

Cluster Relevance. Let $\mathbf{V}^{(i)} \in \mathbb{R}^{L_i \times D}$ denote the retrieval tokens of representative i. We normalize each query and visual token along the feature dimension and compute

$$r_i = \frac{1}{S} \sum_{a=1}^{S} \max_{1 \le b \le L_i} \frac{\mathbf{Q}_a^\top \mathbf{V}_b^{(i)}}{\|\mathbf{Q}_a\|_2 \, \|\mathbf{V}_b^{(i)}\|_2}. \tag{7}$$

Each query token contributes its best visual-token match, and these scores are averaged over the query tokens. The $\min(k, N_c)$ clusters with the highest scores are selected and subsequently assembled in temporal order. The score is a training-free relevance heuristic based on the backbone's existing embeddings.

B.3 Sparse Visual Encoding and Feature Reconstruction

The default selective-reuse path applies the cluster's hot-token mask before the visual transformer. Only input patches corresponding to hot output positions are passed through fresh visual encoding. The resulting features overwrite the corresponding positions in a copy of the cached representative feature map; cold positions retain their cached values. The reconstructed map has the full spatial layout used for LLM input, so token reuse reduces fresh visual computation without itself shortening the reconstructed visual sequence. Image decoding and preprocessing are still required.

Qwen-VL. The mask is defined over visual tokens after spatial merging. For each hot token, the adapter retains its complete group of $s \times s$ input patches, where s is the encoder's spatial merge size. These groups are packed into a compact pseudo-grid with shape $[1, n_{\text{hot}}s, s]$ and passed through the visual encoder, including its transformer blocks and merger. The encoder derives positional information and, where applicable, attention windows from this packed grid. Thus, this path changes the original spatial arrangement seen by the encoder; it does not preserve the dense frame's positional or attention computation. The output hot tokens are then scattered back to their original positions in the representative cache.

LLaVA-OneVision. The adapter uses a fixed spatial cache grid, with 196 positions by default. Each hot grid cell is mapped to the underlying SigLIP patches. Only those patches undergo patch projection, with their original absolute position embeddings retained, followed by the vision-transformer layers needed to produce the configured visual features. The selected features pass through the multimodal projector and are pooled into hot cache-grid cells before reconstruction. Unlike the Qwen pseudo-grid path, this path preserves the selected patches' original position embeddings.

Approximation and Attention. In both sparse paths, cold patches are absent from the current visual-transformer sequence: cached cold features are inserted only after fresh encoding and do

not serve as keys or values in its attention layers. Consequently, the fresh hot features need not equal the corresponding outputs of full-frame encoding, even when cold regions appear unchanged. Selective reuse trades these changes in visual context, and in Qwen's packed spatial layout, for reduced encoding cost; the component ablation measures its resulting accuracy–latency trade-off.

B.4 Gap Summarization and Context Assembly

For a temporal gap $G_j = \{c_l, \ldots, c_r\}$ of non-selected clusters, we initialize $\mathbf{A}_j$ with the first cached representative $\mathbf{e}_{\text{rep}}^{(l)}$ and process the remaining representatives chronologically:

$$\mathbf{A}_j \leftarrow (1-\alpha)\,\mathbf{A}_j + \alpha\,\mathbf{e}_{\text{rep}}^{(i)}, \qquad i = l+1, \ldots, r. \tag{8}$$

The default update rate is $\alpha = 0.1$. EMA weights recent clusters more heavily and represents each gap with a single feature map without encoding additional frames. These summaries are interleaved with selected-cluster features, followed by the recent-frame buffer and query, as described in Section 3.2. They provide coarse cues about omitted intervals but do not recover fine-grained events within those intervals.

C Full Benchmark Results

Table 6 reports the complete benchmark comparison on OVO-Bench and StreamingBench. Table 3 presents the two primary backbones in the main text; this full comparison includes additional model sizes, Qwen3-VL, and training-based reference systems.

The full results show that StreamIndex preserves strong response quality across different backbone families while using the same training-free streaming design. On the `Qwen2.5-VL` family, StreamIndex achieves the best overall score among training-free methods for both model sizes. With `Qwen2.5-VL-7B`, it improves the overall score from 57.69 for vanilla inference with adaptive frame sampling and 59.08 for the strongest training-free baseline to 60.38. The gain is most visible on real-time questions, where StreamIndex reaches 71.84 on OVO-Bench and 74.08 on StreamingBench. With `Qwen2.5-VL-32B`, vanilla inference with adaptive frame sampling and ReKV encounter OOM, while StreamIndex remains feasible and obtains the best overall score of 62.64, outperforming STC and HERMES. This indicates that the proposed index-first design is particularly useful for dynamic-resolution Qwen-style visual encoders, where continuous visual encoding tends to significantly inflate multimodal tokens, ultimately leading to memory bottlenecks.

We also include `Qwen3-VL-8B` as an additional generalization check. StreamIndex improves over vanilla inference with adaptive frame sampling, increasing the overall score from 56.13 to 63.38, which suggests that the proposed mechanism is not tied to a single Qwen generation and can transfer to newer Qwen-VL backbones without training or architecture-specific tuning.

The results on `LLaVA-OV` are more mixed but still support the intended accuracy–scalability trade-off. On `LLaVA-OV-0.5B`, StreamIndex improves the overall score over vanilla inference from 45.87 to 47.82 and obtains the best backward score among the listed training-free variants, although HERMES achieves the strongest overall score. On `LLaVA-OV-7B`, StreamIndex remains close to vanilla inference in overall accuracy (60.45 vs. 60.40) and trails HERMES, which is the strongest accuracy-oriented baseline on this backbone. Combined with the scalability experiments in the main text, these results suggest that StreamIndex does not universally dominate every backbone in raw accuracy, but it maintains competitive response quality while substantially reducing the deployment failures caused by continuous stream-time encoding.

Training-based systems provide an additional reference point, but their objectives differ from the training-free setting targeted by StreamIndex. FlashVStream is designed to improve model performance under a fixed token budget, while Dispider trains the model to decide when to respond during streaming perception. These objectives emphasize different aspects of online video understanding, and their overall benchmark accuracy is therefore lower on the evaluation suite used here. By contrast, StreamForest achieves the highest overall score in Table 6, reflecting the benefit of task-specific training for response quality. Since StreamIndex aims to retrofit existing MLLMs for scalable online deployment without additional model training, the most relevant comparison remains within each open-source backbone family, where StreamIndex offers a favorable balance between benchmark accuracy and deployment robustness.

Table 6: **Bench results.** † denotes adaptive frame sampling based on video length: 1.0 FPS for short videos, 0.5 FPS for medium videos, and 0.2 FPS for long videos (Appendix A).

Methods	Frames Rate	OVO-Bench			StreamingBench	Overall
		Real-time	Backward	Forward	Real-time	
Training-Based						
DISPIDER (CVPR25)	1FPS	55.98	36.42	36.15	68.80	52.90
FLASHVSTREAM (ICCV25)	1FPS	53.16	37.30	7.55	62.12	43.00
STREAMFOREST (NIPS25)	1FPS	67.81	54.13	54.06	75.20	65.19
Training-Free						
LLaVA-OV-0.5B	†	54.99	36.76	36.34	52.92	45.87
+REKV (ICLR25)	1FPS	51.97	34.27	43.86	49.41	43.37
+STC (CVPR26)	1FPS	42.87	24.40	23.73	31.64	28.29
+HERMES	1FPS	54.80	36.63	43.03	62.36	51.56
+STREAMINDEX (Ours)	1FPS	52.07	**39.35**	**43.77**	53.64	47.82
LLaVA-OV-7B	†	63.76	43.86	50.20	72.04	60.40
+REKV (ICLR25)	1FPS	64.60	46.45	47.16	69.52	59.35
+STC (CVPR26)	1FPS	60.31	46.62	47.14	64.20	55.92
+HERMES	1FPS	69.62	48.53	51.08	74.80	63.60
+STREAMINDEX (Ours)	1FPS	65.27	47.24	50.03	70.60	60.45
Qwen2.5-VL-7B	†	60.70	42.48	41.93	72.44	57.69
+REKV (ICLR25)	1FPS	63.65	46.16	39.23	72.84	58.18
+STC (CVPR26)	1FPS	60.66	44.38	35.72	71.00	55.97
+HERMES	1FPS	63.06	46.67	42.90	72.84	59.08
+STREAMINDEX (Ours)	1FPS	**71.84**	46.41	39.53	**74.08**	**60.38**
Qwen2.5-VL-32B	†	-	-	-	-	OOM
+REKV (ICLR25)	1FPS	-	-	-	-	OOM
+STC (CVPR26)	1FPS	66.20	47.46	40.97	72.44	58.68
+HERMES	1FPS	67.21	46.36	50.01	73.08	61.57
+STREAMINDEX (Ours)	1FPS	**74.23**	**47.97**	43.64	**75.88**	**62.64**
Qwen3-VL-8B	†	62.95	39.87	39.52	69.20	56.13
+STREAMINDEX (Ours)	1FPS	73.83	43.80	44.36	77.16	63.38

Category-Level Performance. Figure 7 complements the aggregate scores with fine-grained categories across OVO-Bench and StreamingBench. The profile varies across tasks and backbones, consistent with the main-text observation that improved serving scalability does not imply uniformly higher accuracy on every reasoning task.

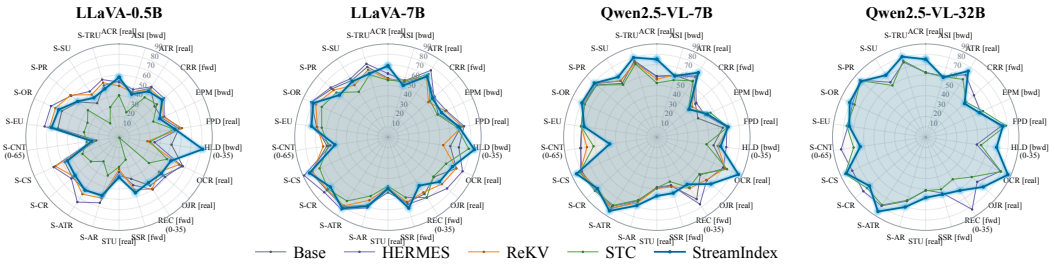


Figure 7: Fine-grained benchmark performance across subtasks in OVO-Bench and StreamingBench.

Table 7: **Composition of the heterogeneous concurrent-stream workload.** Videos are selected from OVO-Bench and retain their native resolutions. The 4/8/12/16-stream settings are nested according to the fixed stream order shown above.

Stream	Used in	Video ID	Native resolution	Duration (s)
1	4/8/12/16	642	854×480	65.0
2	4/8/12/16	1050	1280×720	176.0
3	4/8/12/16	162	1440×1080	590.0
4	4/8/12/16	308	1920×1440	711.0
5	8/12/16	1039	1280×720	67.0
6	8/12/16	1023	1440×1080	187.0
7	8/12/16	1204	1920×1440	177.0
8	8/12/16	667	854×480	316.0
9	12/16	1006	854×480	183.0
10	12/16	1256	1280×720	210.0
11	12/16	1163	1440×1080	301.0
12	12/16	93	1920×1440	301.0
13	16	1144	1440×1080	60.0
14	16	1328	1280×720	299.0
15	16	815	854×480	605.0
16	16	1070	1280×720	609.0

Table 8: **Robustness under heterogeneous concurrent streams.** Each entry reports the number of streams that satisfy the real-time constraint out of the total number of concurrent streams.

Method	4 streams	8 streams	12 streams	16 streams
REKV	3/4	4/8	8/12	6/16
STC	4/4	8/8	5/12	5/16
HERMES	3/4	2/8	1/12	1/16
STREAMINDEX	4/4	8/8	12/12	16/16

D HETEROGENEOUS STREAM EVALUATION

Real-world streaming services rarely receive homogeneous inputs. A deployed system may need to process short clips and long streams at the same time, while the incoming videos may also have substantially different native resolutions. The goal of this experiment is therefore to evaluate whether a streaming method remains stable under *heterogeneous concurrent streams*, where the workload mixes videos with different spatial and temporal costs instead of repeating a single video at a fixed resolution.

We construct this workload from 16 videos selected from OVO-Bench. The selected videos cover four native resolution groups, including 480p, 720p, 1080p, and 1440p, and their durations range from about one minute to nearly twelve minutes. We evaluate four mixed-concurrency settings with 4, 8, 12, and 16 simultaneous streams. The settings are nested: the 4-stream workload uses streams 1–4 in Table 7, the 8-stream workload uses streams 1–8, and so on. This construction keeps the stream order fixed while gradually increasing concurrency, making the scaling behavior directly comparable across settings.

All methods are evaluated with the same video set, stream order, query set, processing frame rate, backbone model, and single-GPU serving configuration described in Appendix A. The goal of this experiment is not to compare fine-grained answer accuracy, but to verify multi-stream robustness under a realistic mixed workload. We therefore summarize robustness by counting how many concurrent streams satisfy the real-time constraint. As shown in Table 8, STREAMINDEX is the only method that keeps all concurrent streams operational under the mixed workload.

Table 9: Sensitivity analysis on `Qwen2.5-VL-7B` varying single parameters from the default ($K_p = 0.8, K_f = 4, m = 4, k = 2$). Total encoded frames include both stream-time and query-time encoding. Query-to-Response Latency measures end-to-end time as defined in Appendix A.

Group	Setting	OVO-Bench Acc. (%)				Total Enc. Frames	Query-to-Resp. Latency (s)
		Real	Bwd.	Fwd.	Total		
Default		71.84	46.41	39.53	52.59	38.33	0.71
K_p	0.4	71.81	44.88	39.14	51.94	40.77	0.90
	0.6	71.64	45.66	39.90	52.40	39.55	0.82
K_f	2	71.45	46.27	39.38	52.37	37.83	0.58
	6	70.99	46.49	39.26	52.25	38.73	0.80
m	2	70.37	45.75	37.28	51.13	36.69	0.64
	6	71.37	46.68	41.24	53.10	39.96	0.79
k	4	70.18	46.81	38.37	51.79	39.41	1.03
	6	68.20	44.68	38.63	50.50	40.39	1.31

E Sensitivity Analysis

We vary one parameter at a time from the default configuration ($K_p = 0.8, K_f = 4, m = 4, k = 2$) on `Qwen2.5-VL-7B`. Table 9 reports OVO-Bench accuracy, total encoded frames, and query-to-response latency, exposing how the recovery and context budgets change the operating point.

Selective Recovery Budgets. Increasing the token reuse ratio K_p from 0.4 to 0.8 reduces latency from 0.90 to 0.71 seconds while total accuracy rises from 51.94% to 52.59% in the evaluated settings. Changing the frame budget K_f from 4 to 2 reduces latency from 0.71 to 0.58 seconds with a small accuracy decrease from 52.59% to 52.37%; increasing it to 6 raises latency to 0.80 seconds without improving accuracy. These measurements support a small frame budget and substantial token reuse for this evaluation, rather than a general claim that processing more visual evidence always improves answers.

Recent Context and Retrieval Budgets. Increasing the recent-frame buffer size m from 2 to 6 improves total accuracy from 51.13% to 53.10%, while increasing latency from 0.64 to 0.79 seconds and total encoded frames from 36.69 to 39.96. Thus, $m = 6$ is a higher-accuracy alternative to the default $m = 4$, with additional query-time cost. Increasing the number of retrieved clusters k from 2 to 6 raises latency from 0.71 to 1.31 seconds and lowers total accuracy from 52.59% to 50.50%. This result is consistent with a larger retrieval budget adding distracting evidence as well as computation, although the table does not isolate the cause of the accuracy decrease. The default configuration balances these measured trade-offs; it is not the highest-accuracy setting for every parameter.

F Limitations and Broader Impacts

F.1 Limitations

StreamIndex is designed for scalable online video understanding under constrained GPU memory and real-time processing budgets, but its performance still depends on the quality of the lightweight stream-time index. The clustering module uses low-cost visual statistics and a fixed distance threshold to decide when to close or extend a cluster. This choice is efficient and robust in our evaluated settings, yet it may under-segment long scenes with subtle state changes or over-segment videos with rapid camera motion, lighting shifts, or visually repetitive backgrounds. Since downstream retrieval operates over these clusters, missed boundaries can hide relevant frames from query-time selection, while overly fragmented clusters can increase retrieval ambiguity and reduce the benefit of compact indexing. Our sensitivity analysis suggests that the default hyperparameters work across a range of settings, but deployment domains with very different video statistics, such as egocentric footage, medical procedures, or dense traffic monitoring, may require retuning the clustering threshold, frame rate, and query-time retrieval budget.

The evaluation also has scope limitations. We focus on training-free instantiations with open-source multimodal large language models and report results on OVO-Bench and StreamingBench using

a single A100 80GB GPU. These benchmarks cover real-time, backward, and forward video QA behavior, but they do not exhaust all long-horizon reasoning patterns, multi-camera dependencies, or safety-critical operating conditions that may arise in practice. In addition, STREAMINDEX reduces stream-time visual encoding rather than eliminating all expensive computation: difficult queries can still require nontrivial query-time encoding, and the final answer quality remains bounded by the underlying MLLM's visual grounding, temporal reasoning, and hallucination behavior. As a result, STREAMINDEX should be viewed as a systems-level mechanism for reallocating computation over time, not as a guarantee of reliable video understanding in domains where errors have high consequences.

F.2 BROADER IMPACTS

STREAMINDEX can make streaming video understanding more accessible by lowering the GPU memory and latency costs of multi-stream deployment, which may benefit applications such as assistive video search, educational video navigation, robotics monitoring, and resource-constrained analytics. At the same time, improving the efficiency of long-running video understanding can also lower barriers to privacy-sensitive or harmful uses, including large-scale surveillance, unauthorized behavioral profiling, and automated monitoring systems that amplify existing biases in the underlying models or deployment data.

F.3 ASSETS AND LICENSES

This work utilizes existing public benchmarks and open-source implementations. For evaluation, we use OVO-Bench and StreamingBench (both under the MIT License) following their official protocols. For model architectures and inference, we employ the `LLaVA-OneVision` and `Qwen-VL` model families, alongside the `SGLang` framework, all of which are distributed under the Apache License 2.0. Furthermore, we compare our approach against several open-source baselines: DISPIDER, FLASHVSTREAM, and STREAMFOREST (Apache License 2.0), as well as HERMES (MIT License). For baselines without explicitly specified licenses (REKV and STC), we strictly adhere to their publicly available code and terms of use. We do not introduce or redistribute any new datasets, pretrained models, or benchmark assets.