RESKV-TREE: RESIDUAL KV-CACHING TREE FOR STREAMING VIDEO UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming video understanding requires storing historical evidence for questions that are unknown when frames arrive, making it essential to reduce memory redundancy while preserving potentially useful details. Existing methods rely on granularity-agnostic storage that compresses all evidence uniformly, making it difficult to preserve information at different granularities. To address this, we propose Residual KV-Caching Tree (**ResKV-Tree**), a hierarchical KV-caching structure for effective and efficient evidence retrieval. During the construction of ResKV-Tree, we select nodes to merge based on semantic similarity, temporal proximity, and evidence saliency, then apply KV non-maximum suppression (KV-NMS) to remove redundant KV pairs using key-based saliency estimation and value-based redundancy suppression. To form the hierarchical structure, we recursively promote salient KV pairs to parent nodes and keep residual ones in children. During question answering, best-first search is employed to explore hierarchical paths starting from the roots and efficiently retrieve relevant evidence at different granularities. Extensive evaluations on two streaming and three offline video understanding benchmarks demonstrate the superiority of ResKV-Tree over baselines and recent methods across three different backbones.

1 INTRODUCTION

Video large language models (Video LLMs) have made substantial progress in understanding pre-recorded videos (Li et al., 2024a; Zhang et al., 2024b; Bai et al., 2025b;a). Streaming video understanding extends this setting to questions that arrive at arbitrary times, requiring answers to be grounded in the video observed so far (Lin et al., 2024b; Li et al., 2025). Because future questions are unknown when frames arrive, details that appear unimportant during encoding may later become essential evidence. Meanwhile, nearby frames and recurring events introduce redundancy within and across video segments. Streaming memory must therefore reduce redundancy while preserving potentially useful details for future questions.

Existing streaming methods maintain visual-feature memories (Yao et al., 2025; Zhang et al., 2025; Zeng et al., 2025) or reuse historical Key-Value pairs through retrieval and compression (Di et al., 2025; Ning et al., 2025; Chen et al., 2026; Xiao et al., 2026; Zhang et al., 2026). Figure 1 illustrates the caching and retrieval challenge with the question: *What color was the lid of the jug that the orange juice was poured into?* Correct answering requires linking the salient juicing and pouring events (KV pairs 2 and 5) to a less salient visual detail: the jug's red lid (KV pair 9). Effective memory management must reduce redundancy while preserving both salient events and complementary fine-grained details for future questions.

To address this challenge, existing methods have explored different memory compression strategies. For example, segment-wise compression methods such as StreamKV (Chen et al., 2026) reduce intra-segment redundancy but leave inter-segment redundancy unaddressed because each segment is compressed independently. The remaining repeated content can consume the retrieval budget and leave necessary details outside the answering context, as illustrated in Fig. 1 (a). On the other hand, progressive aggregation mechanisms (e.g., HERMES (Zhang et al., 2026)) can alleviate inter-segment redundancy by consolidating history into a fixed-size cache. However, repeatedly compressing the accumulated history to fit this budget can sacrifice complementary details needed for future questions, making them unavailable for answering, as illustrated in Fig. 1 (b). Both approaches adopt granularity-

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.

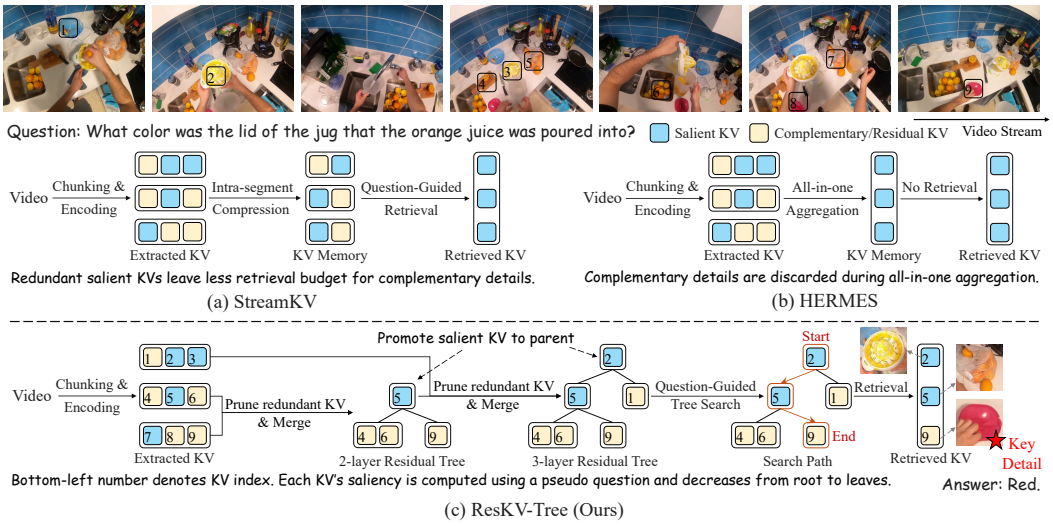

Figure 1: Comparison of KV memory designs for streaming video understanding. Numbers overlaid on the frames denote the locations corresponding to the indexed KV pairs. (a) StreamKV (Chen et al., 2026) compresses segments independently, allowing repeated salient KV pairs to consume the retrieval budget and leave the lid color detail unretrieved. (b) HERMES (Zhang et al., 2026) aggregates history in unified storage, where the lid color detail is discarded rather than retained at a separately accessible level. (c) Our ResKV-Tree builds a hierarchical caching structure by suppressing redundant KV pairs, promoting salient evidence to parents, and retaining complementary details in residual children. Question-guided tree search gathers evidence across levels, combining the salient juicing and pouring evidence with the residual lid color detail to correctly answer *Red*.

agnostic unified storage, without explicitly separating salient evidence and complementary details into different caching levels. These limitations motivate our central question: *How can we construct a hierarchical caching structure that reduces both intra- and inter-segment redundancy and preserves evidence at different granularities to support effective and efficient evidence retrieval?*

Our solution. To address this question, we propose **ResKV-Tree (Residual KV-Caching Tree)**, a hierarchical KV-caching structure for effective and efficient evidence retrieval. During tree construction, ResKV-Tree selects nodes to merge based on semantic similarity, temporal proximity, and evidence saliency. To reduce intra- and inter-segment redundancy, each merge applies KV non-maximum suppression (KV-NMS), combining key-based saliency estimation with value-based redundancy suppression. The retained evidence is then partitioned by promoting salient KV pairs to a parent node and preserving complementary details in residual children, as illustrated in Fig. 1 (c). Repeating this process forms a recursively residual hierarchy that keeps salient evidence closer to the roots and preserves less salient but complementary details in residual children. When a question arrives, ResKV-Tree gathers evidence through question-guided best-first search, starting from the roots and selectively expanding relevant residual children. This search combines salient parent evidence with question-relevant residual details to form a compact cache for question answering.

Our contributions are threefold:

- We introduce ResKV-Tree, a hierarchical KV-caching structure for effective and efficient evidence retrieval. It uses recursive residual partitioning to preserve salient evidence and complementary details at different levels for question-guided retrieval.
- We propose KV-NMS, which combines key-based saliency estimation with value-based redundancy suppression to remove intra- and inter-segment redundancy while preserving complementary evidence.
- Extensive evaluations on two streaming and three offline video benchmarks demonstrate consistent improvements across three different backbones, with ablation studies further validating the effectiveness of our key components.

2 Related Work

Streaming video understanding. Streaming video understanding requires answering questions that can arrive at any time based on the video observed so far. Streaming-specific training supports temporally aligned dialogue and proactive interaction (Chen et al., 2024a; Qian et al., 2025; Wang et al., 2025; Xia et al., 2026), while visual memories, reasoning traces, and latent states extend access to historical context (Zhang et al., 2025; Zeng et al., 2025; Liu et al., 2026; Ge et al., 2026). Training-free alternatives reuse historical KV through question-guided retrieval (Di et al., 2025; Ning et al., 2025; Chen et al., 2026) or maintain a compact cache for direct answering (Zhang et al., 2026; Shen et al., 2026). Our work focuses on efficient KV compression and storage through KV-NMS and a residual KV tree representation, suppressing redundant content while preserving complementary details for question-guided retrieval and reuse.

Streaming memory construction and retrieval. Streaming video understanding relies on a continually updated memory that preserves historical evidence for future questions. LiveVLM (Ning et al., 2025) compresses KV through attention-guided positional bucketing and retrieves pages using position-agnostic keys. MuKV (Xiao et al., 2026) compresses segment-, frame-, and patch-level KV using attention and frequency signals for semi-hierarchical retrieval. StreamKV (Chen et al., 2026) compresses KV within semantic segments for retrieval. HERMES (Zhang et al., 2026) maintains a fixed-size cache through KV pair selection and summary aggregation. These approaches highlight a central challenge in streaming memory: preserving access to fine-grained historical details while limiting redundant evidence during retrieval. To address this, ResKV-Tree organizes salient and complementary KV pairs into non-overlapping parent and child sets, enabling question-guided retrieval of both representative evidence and fine-grained details.

3 Method

ResKV-Tree is a hierarchical KV-caching structure for effective and efficient evidence retrieval in streaming video understanding. As shown in Figure 2, ResKV-Tree has two core components: Residual KV Tree Construction and Question-Guided Tree Search.

3.1 Overall Pipeline

As illustrated in Fig. 2 (a), we first encode incoming video chunks into KV pairs using a frozen video language model and compute their saliency scores with a fixed pseudo question. During Residual KV Tree Construction (see Fig. 2 (b)), each layer organizes historical KV pairs into residual KV trees by recursively selecting roots to merge, suppressing redundant KV pairs, and partitioning retained evidence between a salient parent and residual children. Figure 3 details the merge criteria and KV-NMS operator. When a user question arrives, Question-Guided Tree Search retrieves relevant KV pairs through a shared priority queue under a historical KV budget, as shown in Fig. 2 (c). The retrieved KV pairs are assembled with the text prefix and latest chunk for answer generation.

Video chunking and encoding. We divide the observed video into chunks of F sampled frames for KV extraction. For every Transformer layer ℓ, we extract keys before rotary position embedding (RoPE) and their corresponding values. Storing pre-RoPE keys enables semantic retrieval without interference from positional embeddings, and the positional embeddings will be reapplied to the retrieved keys before final question answering. Each layer maintains its own residual KV trees $\mathcal{T}_\ell$, allowing merge decisions and retrieved evidence to vary across layers.

KV saliency computation. After video encoding, we compute KV saliency using a fixed prompt *What happens in the video?* as the pseudo question. For a text query x, a forward pass conditioned on the fixed prefix yields pre-RoPE queries, which are averaged over its text tokens to obtain $\bar{\mathbf{q}}_{\ell,h}$ for query head h. The saliency of KV pair i is computed by averaging query-key cosine similarities across query heads:

$$s_{\ell,i} = \frac{1}{H_q} \sum_{h=1}^{H_q} \cos(\bar{\mathbf{q}}_{\ell,h}, \mathbf{k}_{\ell,h,i}) \,, \tag{1}$$

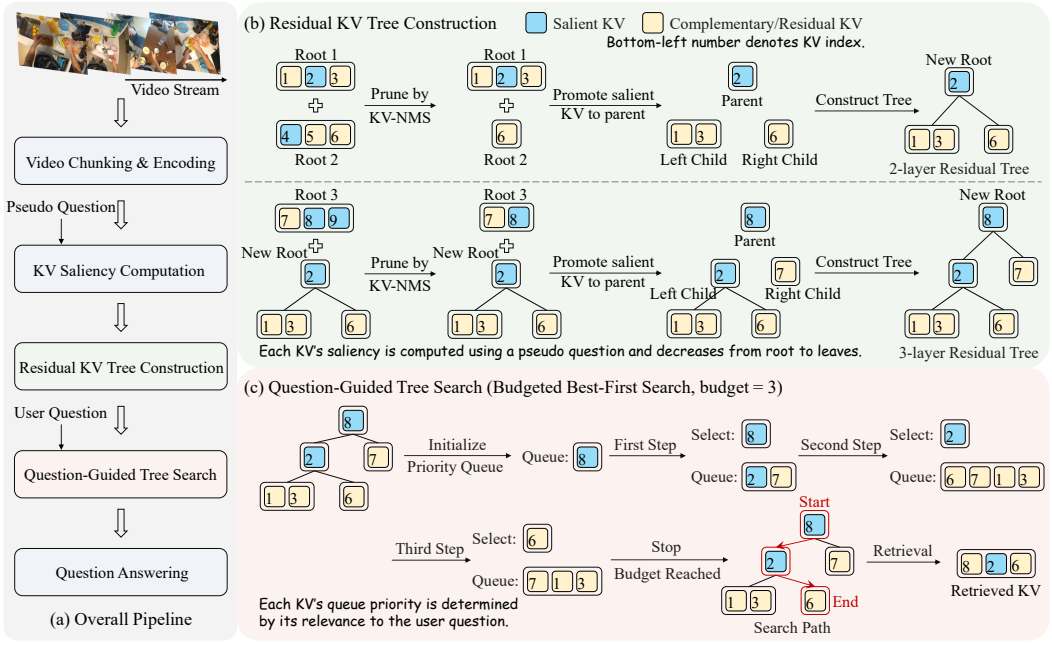

Figure 2: Overview of ResKV-Tree for hierarchical KV caching and evidence retrieval. (a) The pipeline consists of five main steps. A pseudo question provides saliency scores for residual tree construction, while the user question guides evidence retrieval for answer generation. (b) Each merge applies KV-NMS to the selected roots, promotes salient KV pairs to a new parent, and retains complementary KV pairs in residual children. (c) Question-guided best-first search uses a priority queue to retrieve KV pairs, with priorities determined by their relevance to the user question, and stops when the budget is reached.

where H_q is the number of query heads and $\mathbf{k}_{\ell,h,i}$ is the key of KV pair i corresponding to query head h in layer ℓ. For simplicity, we omit the layer index ℓ and write s_i for $s_{\ell,i}$ in the content below.

3.2 Residual KV Tree Construction

With the extracted KV pairs and their saliency scores, we build residual KV trees through recursive merging. Fig. 2 (b) illustrates two successive merges: the first forms a parent with residual children, and the second merges this root with another root while preserving the existing descendants. Each merge combines two core operations, KV-NMS pruning and salient KV promotion, to form the residual tree structure. The roots participating in a merge are selected using the criteria in Fig. 3 (a), and their current KV pairs are pruned using Fig. 3 (b).

Merge pair selection. Each layer maintains at most N trees. Whenever inserting a leaf exceeds this limit, we select two roots to merge using the criteria in Fig. 3 (a). Let $\mathcal{I}_n$ index the KV pairs currently held by node n, and let $\mathbf{v}_i$ concatenate the values of KV pair i across KV heads. A node's semantic descriptor and mean saliency are $\mathbf{z}_n = |\mathcal{I}_n|^{-1} \sum_{i\in\mathcal{I}_n} \mathbf{v}_i$ and $\bar{s}_n = |\mathcal{I}_n|^{-1} \sum_{i\in\mathcal{I}_n} s_i$. For a node spanning the frame interval $[a_n, b_n)$, its temporal midpoint is normalized over the current roots as $\tilde{t}_n = ((a_n + b_n)/2 - a_{\min})/\max(1, b_{\max} - a_{\min})$, where $a_{\min}$ and $b_{\max}$ are their earliest start and latest end. For a candidate pair (n, m), we minimize

$$J(n,m) = \lambda_D\big(1 - \cos(\mathbf{z}_n, \mathbf{z}_m)\big) + \lambda_T \exp\left(\frac{|\tilde{t}_n - \tilde{t}_m|}{\tau}\right) + \lambda_S \frac{\bar{s}_n + \bar{s}_m}{2}, \tag{2}$$

where $\lambda_D, \lambda_T, \lambda_S$ balance the three criteria and τ controls temporal sensitivity. This favors merging semantically similar, temporally close, and less salient roots, keeping distinctive or highly salient evidence closer to the roots for earlier retrieval. Note that root pairs from temporally non-adjacent chunks may also be selected.

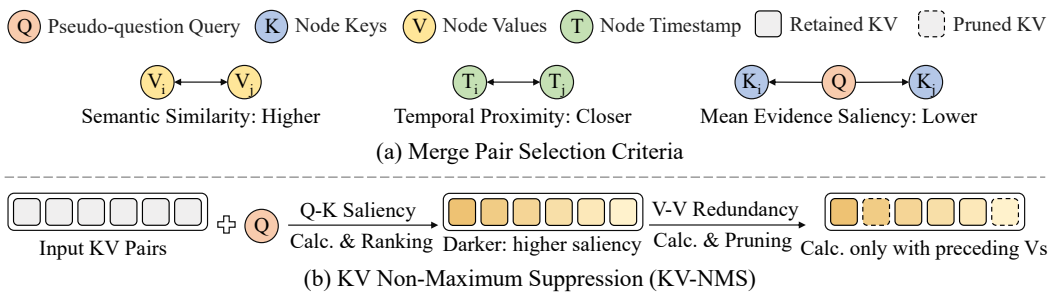

Figure 3: (a) Merge pair selection criteria, which favor roots with higher value-based semantic similarity, closer temporal positions, and lower mean pseudo-question saliency. (b) KV pairs are ranked by Q-K saliency, with darker shading indicating higher scores. Each value is compared only with preceding values in this order to compute V-V redundancy. KV pairs with the highest redundancy are pruned to meet the retention target.

KV non-maximum suppression. We pool the L KV pairs held by the selected roots and apply KV-NMS to suppress intra- and inter-chunk redundancy (Fig. 3 (b)). Let π sort these pairs by descending pseudo-question saliency s_i. The redundancy of each pair is its maximum value similarity to preceding pairs:

$$d_r = \max_{u<r} \cos\big(\mathbf{v}_{\pi(r)}, \mathbf{v}_{\pi(u)}\big) \quad (r > 1), \qquad d_1 = -\infty. \tag{3}$$

We discard the $\lfloor \alpha L \rfloor$ pairs with the highest redundancy scores, retaining $L' = L - \lfloor \alpha L \rfloor$ pairs, where α is a hyperparameter controlling the pruning ratio. Saliency thus determines the comparison order, while value redundancy determines removal, allowing less salient but complementary KV pairs to survive. Algorithm 2 in Appendix C details the KV-NMS procedure.

Salient promotion and residual retention. Let $\mathcal{I}_r$ be the index set retained by KV-NMS, with size L'. At each merge, the $\lfloor \beta L' \rfloor$ retained KV pairs with the highest pseudo-question saliency form the parent set $\mathcal{P}$, where β is a hyperparameter controlling the promotion ratio. The remaining retained KV pairs return to their source nodes as residual sets:

$$\mathcal{R}_n = (\mathcal{I}_r \cap \mathcal{I}_n) \setminus \mathcal{P}, \qquad \mathcal{R}_m = (\mathcal{I}_r \cap \mathcal{I}_m) \setminus \mathcal{P}. \tag{4}$$

Here, $\mathcal{I}_r \cap \mathcal{I}_n$ selects the KV pairs from node n that survive KV-NMS, and subtracting $\mathcal{P}$ removes those promoted to the parent, leaving the residual set $\mathcal{R}_n$. The same operation defines $\mathcal{R}_m$ for node m. The new parent replaces the two roots, which become its residual children and retain their existing descendants. This partition defines the recursively residual organization of ResKV-Tree, with salient KV pairs in parents and complementary KV pairs in residual children, stored without duplication and separately retrievable. In Fig. 2 (b), the first merge prunes KV pairs 4 and 5, promotes KV pair 2, and leaves $\{1, 3\}$ and 6 in the residual children. The next merge prunes KV pair 9 and promotes KV pair 8, leaving KV pairs 2 and 7 as residual children while preserving the descendants of KV pair 2. The complete residual KV tree construction procedure is provided in Algorithm 1 in Appendix C.

3.3 Question-Guided Tree Search

The residual trees organize salient KV pairs in parent nodes and complementary evidence in residual children. To retrieve question-relevant evidence within a historical KV budget B, we perform question-guided tree search. Given a user question x, we compute the relevance of each KV pair to x using Equation 1 and score each node by averaging these relevance scores.

Each layer performs an independent best-first search, with all roots initially placed in a shared priority queue. At each step, we select the highest-scoring node and collect its KV pairs. If the accumulated count is below B, we score its children and insert them into the queue. The search stops when the count reaches B or the queue is empty. This procedure allows question-relevant residual evidence to take priority over other queued nodes, combining salient parent KV pairs with complementary details relevant to the question. After retrieval, we sort the selected KV pairs by their source temporal

Table 1: Accuracy comparison on two streaming benchmarks: StreamingBench and OVO-Bench. Real-Time and Backward denote Real-Time Visual Perception and Backward Tracing, respectively. Best results within each backbone group are in bold.

Model	StreamingBench	OVO-Bench		
		Real-Time	Backward	Avg.
Offline MLLMs				
InternVL-V2-8B (Chen et al., 2024b)	63.72	60.73	44.00	52.37
Qwen2-VL-7B (Wang et al., 2024)	69.04	60.65	48.58	54.62
LLaVA-Video-7B (Zhang et al., 2024b)	–	63.34	41.68	52.51
LongVU-7B (Shen et al., 2024)	–	57.40	39.49	48.45
Training-based Online MLLMs				
VideoLLM-online-8B (Chen et al., 2024a)	35.99	20.79	17.73	19.26
Flash-VStream-7B (Zhang et al., 2025)	23.23	29.86	25.35	27.61
Dispider-7B (Qian et al., 2025)	67.63	54.55	36.06	45.31
TimeChat-Online-7B (Yao et al., 2025)	75.36	61.90	41.70	51.80
StreamBridge-7B (Wang et al., 2025)	77.04	71.30	–	–
StreamForest-7B (Zeng et al., 2025)	77.26	61.20	52.02	56.61
Streamo-7B (Xia et al., 2026)	–	65.98	46.10	56.04
ThinkStream-3B (Liu et al., 2026)	75.00	67.03	52.30	59.66
Training-free Offline-to-Online Methods				
LLaVA-OV-7B (Li et al., 2024a)	71.34	63.06	43.64	53.35
+ ReKV (Di et al., 2025)	69.22	57.33	44.16	50.75
+ LiveVLM (Ning et al., 2025)	72.92	–	–	–
+ StreamKV (Chen et al., 2026)	68.80	–	–	–
+ MuKV (Xiao et al., 2026)	64.40	–	–	–
+ HERMES (Zhang et al., 2026)	73.23	66.34	50.20	58.27
+ **Ours**	**74.19**	**78.03**	**50.27**	**64.15**
Qwen2.5-VL-7B (Bai et al., 2025b)	73.31	59.90	44.65	52.28
+ HERMES (Zhang et al., 2026)	79.44	68.98	49.43	59.21
+ **Ours**	**80.48**	**78.94**	**53.19**	**66.06**
Qwen3-VL-8B (Bai et al., 2025a)	75.66	68.21	46.42	57.32
+ HERMES (Zhang et al., 2026)	81.28	73.29	49.28	61.29
+ **Ours**	**82.20**	**81.78**	**56.30**	**69.04**

coordinates and reapply RoPE to the keys using compact positions reconstructed from their source coordinates. These KV pairs are then assembled with the user question to form a compact KV cache for answer generation. Algorithm 3 in Appendix C details the retrieval procedure.

4 Experiments

4.1 Experimental Setup

We evaluate ResKV-Tree on both streaming and offline video understanding benchmarks using LLaVA-OneVision-7B (Li et al., 2024a), Qwen2.5-VL-7B (Bai et al., 2025b), and Qwen3-VL-8B (Bai et al., 2025a) as backbones. For streaming video understanding, we use the Real-Time Visual Understanding subset of StreamingBench (Lin et al., 2024b) and the Real-Time Visual Perception and Backward Tracing subsets of OVO-Bench (Li et al., 2025). At each question timestamp, the model can access only the video content observed up to that point. For offline video understanding, we use VideoMME (Fu et al., 2025), MLVU (Zhou et al., 2025), and MVBench (Li et al., 2024b) to assess temporal understanding across long and short videos. We sample videos at 1 FPS and process them in 16-frame chunks. Additional hyperparameters are provided in Appendix B.1.

4.2 Main Results

Streaming video understanding. Table 1 compares ResKV-Tree with offline MLLMs, training-based online MLLMs, and training-free offline-to-online methods on StreamingBench and OVO-Bench, covering both current-scene understanding and reasoning over previously observed content.

Table 2: Accuracy comparison on three offline video benchmarks: VideoMME, MLVU, and MVBench. Best results within each backbone group are in bold.

Model	VideoMME	MLVU	MVBench	Avg.
Offline MLLMs				
InternVL-V2-8B (Chen et al., 2024b)	54.0	48.1	66.4	56.2
LongVA-7B (Zhang et al., 2024a)	52.4	56.3	49.2	52.6
Qwen2-VL-7B (Wang et al., 2024)	63.3	64.8	67.0	65.0
LongVU-7B (Shen et al., 2024)	60.6	65.4	66.9	64.3
LLaVA-Video-7B (Zhang et al., 2024b)	63.3	70.8	58.6	64.2
Training-based Online MLLMs				
VideoChat-Online-4B (Huang et al., 2025)	54.4	60.8	65.2	60.1
Dispider-7B (Qian et al., 2025)	57.2	61.7	–	–
TimeChat-Online-7B (Yao et al., 2025)	62.4	62.6	–	–
StreamBridge-7B (Wang et al., 2025)	64.4	69.6	64.4	66.1
StreamForest-7B (Zeng et al., 2025)	61.4	70.0	70.2	67.2
Streamo-7B (Xia et al., 2026)	67.9	–	72.3	–
Training-free Offline-to-Online Methods				
LLaVA-OV-7B (Li et al., 2024a)	58.2	64.7	56.7	59.9
+ ReKV (Di et al., 2025)	57.7	68.5	56.8	61.0
+ LiveVLM (Ning et al., 2025)	59.6	68.1	–	–
+ MuKV (Xiao et al., 2026)	61.2	67.8	–	–
+ HERMES (Zhang et al., 2026)	58.9	68.0	56.9	61.3
+ **Ours**	**61.8**	**68.8**	**57.2**	**62.6**
Qwen2.5-VL-7B (Bai et al., 2025b)	64.5	66.9	65.0	65.5
+ HERMES (Zhang et al., 2026)	60.6	67.4	65.5	64.5
+ **Ours**	**65.9**	**69.1**	**68.6**	**67.9**
Qwen3-VL-8B (Bai et al., 2025a)	70.5	75.8	68.2	71.5
+ **Ours**	**72.0**	**77.6**	**69.9**	**73.2**

Across these evaluations, ResKV-Tree consistently improves over the corresponding base models and outperforms the listed training-free methods using the same backbone. Moreover, its Qwen-based variants even outperform the listed training-based online models on both benchmarks. These results demonstrate the effectiveness of ResKV-Tree in organizing and retrieving existing KV pairs to improve streaming video understanding across different backbones. A detailed breakdown of these results by task is provided in Appendix A.

Offline video understanding. To examine whether these benefits extend beyond streaming, we further evaluate ResKV-Tree on VideoMME, MLVU, and MVBench, which cover offline understanding of long and short videos. As shown in Table 2, ResKV-Tree consistently improves over the corresponding base models on all three benchmarks. In particular, the gains on VideoMME and MLVU highlight the value of retaining complementary evidence from video history and retrieving it according to the question, while the improvements on MVBench show that the same memory organization remains effective for short video understanding. Together with the streaming results, these consistent gains demonstrate the robustness and generalizability of ResKV-Tree across different evaluation settings.

Efficiency analysis. Figure 4 compares the accuracy-efficiency trade-off between ResKV-Tree and representative streaming methods on LLaVA-OV-7B, with measurements taken after processing 1024 frames in a streaming manner. TTFT (time to first token) measures the latency from query arrival to the generation of the first answer token. ResKV-Tree achieves the highest accuracy while maintaining sub-second TTFT. Compared with HERMES, it improves accuracy by 5.88 percentage points with a slightly longer TTFT and a small increase in peak GPU memory usage. These results demonstrate the value of retaining complementary residual evidence in a searchable tree while retrieving a compact set of KV pairs for low-latency generation.

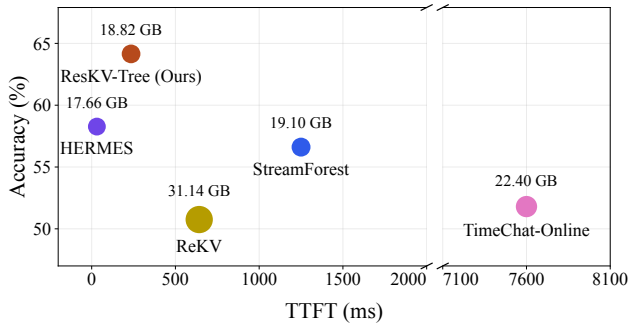

Figure 4: Accuracy-efficiency comparison on OVO-Bench. The horizontal and vertical axes show time to first token (TTFT) and accuracy, respectively. Annotations indicate peak GPU memory usage.

4.3 Ablation Studies

Residual retention and tree search. Residual mechanism preserves complementary KV pairs in child nodes after salient KV pairs are promoted to parents, while question-guided tree search retrieves relevant evidence from this memory for answering. We examine their individual and joint contributions by selectively enabling each component. When the residual mechanism is disabled, we fuse KV pairs using ToMe-style bipartite soft matching (Bolya et al., 2022) based on key similarity, separately averaging the keys and the values within each matched group. To ablate tree search, we select all root nodes and use only their KV pairs for answering. As shown in Table 3, enabling either component improves accuracy over these vanilla implementations across all three benchmarks, and combining them consistently yields the best performance. These results demonstrate the advantage of our residual retention and question-guided tree search design.

Table 3: Ablation study on the key components of ResKV-Tree with Qwen2.5-VL-7B.

Residual Mechanism	Tree Search	StreamingBench	OVO-Bench	MLVU
–	–	78.3	62.2	65.6
–	✓	79.1	63.4	66.9
✓	–	79.9	64.7	68.0
✓	✓	**80.5**	**66.1**	**69.1**

Merge pair selection. Merge pair selection combines spatio-temporal redundancy and saliency to determine which roots to consolidate. Spatio-temporal redundancy favors semantically similar and temporally close roots, while saliency prioritizes less salient roots for merging. We examine their contributions by selectively enabling each criterion. As shown in Table 4, either criterion improves accuracy across the evaluated benchmarks, with spatio-temporal redundancy providing the larger individual gain. Combining them consistently achieves the best performance. These results show the benefit of consolidating related evidence while keeping salient roots independently accessible, supporting the effectiveness of our hierarchical KV-cache construction method for streaming videos.

Table 4: Ablation study on merge strategies of ResKV-Tree with Qwen2.5-VL-7B.

Spatio-Temporal Redundancy	Saliency	StreamingBench	OVO-Bench	MLVU
–	–	79.7	65.0	67.7
–	✓	80.1	65.3	68.1
✓	–	80.4	65.8	68.7
✓	✓	**80.5**	**66.1**	**69.1**

Effect of KV-NMS. KV-NMS combines saliency-based ordering with value-based redundancy suppression to select KV pairs at each merge. We compare it with random selection, saliency-only selection, and redundancy-only selection under the same retention ratio. Random selection uniformly samples KV pairs, while saliency-only selection retains those with the highest pseudo-question Q-K cosine similarity. Redundancy-only selection randomly orders KV pairs and removes those

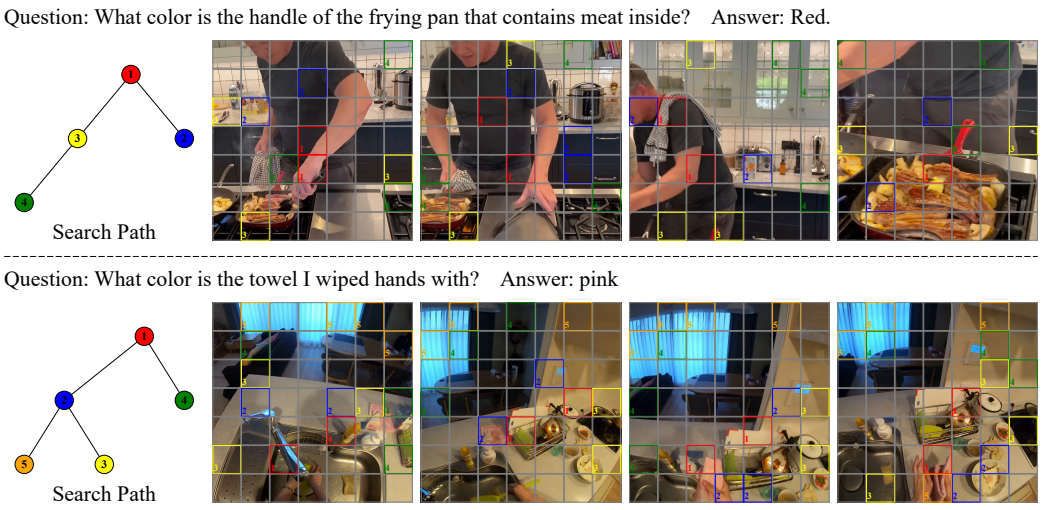

Figure 5: Two visualization examples of question-guided retrieval in ResKV-Tree. Node numbers indicate the visitation order during best-first search. Patch labels identify the nodes that store the KV pairs corresponding to those spatial locations.

with the highest maximum V-V cosine similarity to preceding pairs, whereas KV-NMS applies this suppression rule after ordering pairs by saliency. As shown in Table 5, saliency-only and redundancy-only selection each outperform random selection across the evaluated benchmarks, and KV-NMS consistently achieves the highest accuracy. These results demonstrate the benefit of jointly considering saliency and redundancy for KV pair retention.

Table 5: Ablation study on KV-NMS strategies of ResKV-Tree with Qwen2.5-VL-7B.

Method	StreamingBench	OVO-Bench	MLVU
Random	78.7	64.0	63.3
Saliency-only	79.8	65.1	67.5
Redundancy-only	79.2	65.2	66.9
Ours	**80.5**	**66.1**	**69.1**

4.4 Visualizations

Figure 5 presents two visualization examples, each showing four randomly sampled frames and the retrieval path through a randomly selected tree. For clarity, each frame is presented as a 7×7 grid. In the first example, residual nodes retain details such as the red pan handle, complementing the salient evidence stored in parent nodes. In the second example, the retrieved evidence links the towel to the hand movements across frames. Together, these examples illustrate how recursively residual storage preserves complementary evidence for question-guided retrieval. Additional examples are presented in Appendix D.

5 Concluding Remarks

We propose Residual KV-Caching Tree (ResKV-Tree), a hierarchical KV-caching structure for effective and efficient evidence retrieval. ResKV-Tree organizes historical KV pairs into a residual hierarchy, where salient KV pairs in parents and complementary KV pairs in children form non-overlapping sets. KV non-maximum suppression combines key-based saliency with value-based similarity to reduce redundancy within and across video segments. When a question arrives, question-guided hierarchical retrieval assembles representative evidence and relevant residual details into a compact answering cache. Experiments on two streaming and three offline video understanding benchmarks validate the efficacy of our approach, demonstrating consistent performance improvements across three different backbones.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code implementation. We did not use generative AI tools for method conceptualization, hypothesis formulation, experimental design, or result interpretation. Synthetic dataset generation, dataset cleaning and reformatting, qualitative and thematic data analysis, and mathematical proof development are not applicable to this work. Additionally, we used generative AI tools for language editing and LaTeX formatting. The authors reviewed and tested the AI-assisted code, reviewed the manuscript text, and verified references against the original sources. All authors take responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our study does not involve human subjects, personal data, or sensitive attributes, and thus does not raise direct privacy or security concerns. The datasets used are publicly available, and we strictly follow their licenses and usage guidelines. No proprietary or non-consensually collected data were employed. We declare no conflicts of interest and ensure compliance with ethical standards in data handling and research integrity.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to ensure the reproducibility of ResKV-Tree. Sections 3 and 4.1 of the main paper describe the residual KV tree construction and question-guided retrieval procedures, backbone models, evaluation benchmarks, and video sampling settings. Additional implementation details are provided in Appendix B. We will release the code following the completion of the blind review process.

REFERENCES

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, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923, 2025b.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461, 2022.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18407–18418, 2024a.

Yilong Chen, Xiang Bai, Zhibin Wang, Chengyu Bai, Yuhan Dai, and Ming Lu. Streamkv: Streaming video question-answering with segment-based kv cache retrieval and compression. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 40, pp. 3120–3128, 2026.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 24185–24198, 2024b.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. arXiv preprint arXiv:2406.07476, 2024.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video kv-cache retrieval. In International Conference on Learning Representations, volume 2025, pp. 42115–42127, 2025.

Jiajun Fei, Dian Li, Zhidong Deng, Zekun Wang, Gang Liu, and Hui Wang. Video-ccam: Enhancing video-language understanding with causal cross-attention masks for short and long videos. arXiv preprint arXiv:2408.14023, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 24108–24118, 2025.

Haonan Ge, Yiwei Wang, Hang Wu, and Yujun Cai. What should a streaming video model remember? arXiv preprint arXiv:2606.16353, 2026.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. arXiv preprint arXiv:2404.06395, 2024.

Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 3328–3338, 2025.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. arXiv preprint arXiv:2408.03326, 2024a.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024b.

Yifei Li, Junbo Niu, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, et al. Ovo-bench: How far is your video-llms from real-world online video understanding? arXiv preprint arXiv:2501.05510, 2025.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 26689–26699. IEEE, 2024a.

Junming Lin, Zheng Fang, Chi Chen, Zihao Wan, Fuwen Luo, Peng Li, Yang Liu, and Maosong Sun. Streamingbench: Assessing the gap for mllms to achieve streaming video understanding. arXiv preprint arXiv:2411.03628, 2024b.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, january 2024. URL https://llava-vl.github. io/blog/2024-01-30-llava-next, 1(8), 2024a.

Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input. arXiv preprint arXiv:2408.15542, 2024b.

Zikang Liu, Longteng Guo, Handong Li, Ru Zhen, Xingjian He, Ruyi Ji, Xiaoming Ren, Yanhao Zhang, Haonan Lu, and Jing Liu. Thinking in streaming video. arXiv preprint arXiv:2603.12938, 2026.

Zhenyu Ning, Guangda Liu, Qihao Jin, Chengwei Li, Wenchao Ding, Minyi Guo, and Jieru Zhao. Livevlm: Efficient online video understanding via streaming-oriented kv cache and retrieval. arXiv preprint arXiv:2505.15269, 2025.

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. arXiv preprint arXiv:2501.03218, 2025.

Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. Longvu: Spatiotemporal adaptive compression for long video-language understanding. arXiv preprint arXiv:2410.17434, 2024.

Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. arXiv preprint arXiv:2604.02317, 2026.

Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. Streambridge: Turning your offline video large language model into a proactive streaming assistant. arXiv preprint arXiv:2505.05467, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191, 2024.

Jiaer Xia, Peixian Chen, Mengdan Zhang, Xing Sun, and Kaiyang Zhou. Streaming video instruction tuning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 31219–31229, 2026.

Junbin Xiao, Jiajun Chen, Tianxiang Sun, Xun Yang, and Angela Yao. Mukv: Multi-grained kv cache compression for long streaming video question-answering. arXiv preprint arXiv:2605.22269, 2026.

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. arXiv preprint arXiv:2504.17343, 2025.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, et al. Streamforest: Efficient online video understanding with persistent event memory. Advances in Neural Information Processing Systems, 38:75804–75835, 2025.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-vstream: Efficient real-time understanding for long video streams. arXiv preprint arXiv:2506.23825, 2025.

Haowei Zhang, Shudong Yang, Jinlan Fu, See Kiong Ng, and Xipeng Qiu. Hermes: Kv cache as hierarchical memory for efficient streaming video understanding. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8411–8430, 2026.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. arXiv preprint arXiv:2406.16852, 2024a.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. arXiv preprint arXiv:2410.02713, 2024b.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13691–13701, 2025.

ResKV-Tree: Residual KV-Caching Tree For Streaming Video Understanding

Supplementary Material

A Full Results on StreamingBench and OVO-Bench

A.1 StreamingBench

StreamingBench (Lin et al., 2024b) evaluates streaming video understanding across three aspects: real-time visual understanding, omni-source understanding, and contextual understanding. We evaluate on its Real-Time Visual Understanding subset, which contains 2,500 questions across 500 videos and covers 10 tasks. Table 6 reports the full results for all the 10 tasks.

As shown in Table 6, ResKV-Tree achieves the highest average accuracy among the compared methods using each backbone and attains the best results on most of the ten tasks within each backbone group. Its Qwen3-VL-8B variant reaches 82.20% average accuracy, outperforming all listed methods, including training-based online MLLMs. The gains span visual perception, summarization, and reasoning, with particularly strong results in object perception, clips summarization, text-rich understanding, and spatial understanding. These results demonstrate the effectiveness of recursively organizing salient and complementary KV pairs and retrieving question-relevant evidence for diverse streaming understanding tasks.

Table 6: Accuracy comparison on *Real-Time Visual Understanding* task of StreamingBench, which consists of Object Perception (OP), Causal Reasoning (CR), Clips Summarization (CS), Attribute Perception (ATP), Event Understanding (EU), Text-Rich Understanding (TR), Prospective Reasoning (PR), Spatial Understanding (SU), Action Perception (ACP), and Counting (CT). Best results within each backbone group are in bold.

Model	OP	CR	CS	ATP	EU	TR	PR	SU	ACP	CT	Avg.
Offline MLLMs											
Video-LLaMA2-7B (Cheng et al., 2024)	55.86	55.47	57.41	58.17	52.80	43.61	39.81	42.68	45.61	35.23	49.52
VILA-1.5-8B (Lin et al., 2024a)	53.68	49.22	70.98	56.86	53.42	53.89	54.63	48.78	50.14	17.62	52.32
Video-CCAM-14B (Fei et al., 2024)	56.40	57.81	65.30	62.75	64.60	51.40	42.59	47.97	49.58	31.61	53.96
LongVA-7B (Zhang et al., 2024a)	70.03	63.28	61.20	70.92	62.73	59.50	61.11	53.66	54.67	34.72	59.96
InternVL-V2-8B (Chen et al., 2024b)	68.12	60.94	69.40	77.12	67.70	62.93	59.26	53.25	54.96	56.48	63.72
Kangaroo-7B (Liu et al., 2024b)	71.12	84.38	70.66	73.20	67.08	61.68	56.48	55.69	62.04	38.86	64.60
LLaVA-NeXT-Video-32B (Liu et al., 2024a)	78.20	70.31	73.82	76.80	63.35	69.78	57.41	56.10	64.31	38.86	66.96
MiniCPM-V-2.6-8B (Hu et al., 2024)	71.93	71.09	77.92	75.82	64.60	65.73	70.37	56.10	62.32	53.37	67.44
Qwen2-VL-7B (Wang et al., 2024)	75.20	82.81	73.19	77.45	68.32	71.03	72.22	61.19	61.47	46.11	69.04
Training-based Online MLLMs											
VideoLLM-online-8B (Chen et al., 2024a)	39.07	40.06	34.49	31.05	45.96	32.40	31.48	34.16	42.49	27.89	35.99
Flash-VStream-7B (Zhang et al., 2025)	25.89	43.57	24.91	23.87	27.33	13.08	18.52	25.20	23.87	48.70	23.23
Dispider-7B (Qian et al., 2025)	74.92	75.53	74.10	73.08	74.44	59.92	76.14	62.91	62.16	45.80	67.63
TimeChat-Online-7B (Yao et al., 2025)	80.22	82.03	79.50	83.33	76.10	78.50	78.70	64.63	69.60	57.98	75.36
StreamBridge-7B (Wang et al., 2025)	84.74	82.68	88.92	89.77	77.36	85.36	84.26	69.92	71.67	35.75	77.04
StreamForest-7B (Zeng et al., 2025)	83.11	82.81	82.65	84.26	77.50	78.19	76.85	69.11	75.64	54.40	77.26
ThinkStream-3B (Liu et al., 2026)	83.74	70.31	76.03	82.69	76.73	78.19	76.85	70.73	74.43	45.21	75.00
Training-free Offline-to-Online Methods											
LLaVA-OV-7B (Li et al., 2024a)	78.75	78.12	80.76	81.19	71.70	72.59	72.22	63.82	66.01	38.34	71.34
+ ReKV (Di et al., 2025)	76.02	81.25	77.92	76.90	66.04	66.04	69.44	60.98	64.31	**49.22**	69.22
+ LiveVLM (Ning et al., 2025)	81.47	78.13	83.28	79.08	69.57	**74.14**	75.00	69.11	67.71	40.41	72.92
+ StreamKV (Chen et al., 2026)	73.80	77.30	85.90	77.50	73.30	63.90	69.40	61.40	63.20	35.80	68.80
+ MuKV (Xiao et al., 2026)	74.00	78.20	72.20	71.80	64.80	55.50	66.70	58.90	59.40	39.40	64.40
+ HERMES (Zhang et al., 2026)	79.02	81.25	**87.70**	80.20	69.18	71.96	73.15	66.26	69.41	43.52	73.23
+ **Ours**	**82.02**	**84.38**	83.91	**81.85**	**74.84**	67.91	**80.56**	**73.17**	**69.97**	39.90	**74.19**
Qwen2.5-VL-7B (Bai et al., 2025b)	77.93	76.56	78.55	80.86	76.73	76.95	80.56	65.45	65.72	**52.85**	73.31
+ HERMES (Zhang et al., 2026)	83.65	**81.25**	88.01	**87.46**	76.73	86.60	82.41	76.02	73.94	46.63	79.44
+ **Ours**	**87.19**	78.13	**89.91**	86.14	**83.02**	**87.23**	**85.19**	**76.42**	**75.07**	44.04	**80.48**
Qwen3-VL-8B (Bai et al., 2025a)	78.32	75.00	78.55	84.62	70.44	80.25	83.33	71.54	70.45	**51.06**	75.66
+ HERMES (Zhang et al., 2026)	84.47	**79.69**	89.91	88.78	76.73	88.16	**87.04**	76.83	**79.04**	49.22	81.28
+ **Ours**	**88.01**	78.91	**91.48**	**90.76**	**81.76**	**90.34**	**87.04**	**78.86**	74.50	47.15	**82.20**

Table 7: Accuracy comparison on *Real-Time Visual Perception* and *Backward Tracing* tasks of OVO-Bench. Real-Time Visual Perception task consists of Optical Character Recognition (OCR), Action Recognition (ACR), Attribute Recognition (ATR), Spatial Understanding (STU), Future Prediction (FPD), and Object Recognition (OJR). Backward Tracing task consists of Episodic Memory (EPM), Action Sequence Identification (ASI), and Hallucination Detection (HLD). Best results within each backbone group are in bold.

Model	Real-Time Visual Perception							Backward Tracing				Overall
	OCR	ACR	ATR	STU	FPD	OJR	Avg.	EPM	ASI	HLD	Avg.	Avg.
Offline MLLMs												
InternVL-V2-8B (Chen et al., 2024b)	68.46	58.72	68.97	44.94	67.33	55.98	60.73	43.10	61.49	27.41	44.00	52.37
Qwen2-VL-7B (Wang et al., 2024)	69.13	53.21	63.79	50.56	66.34	60.87	60.65	44.44	66.89	34.41	48.58	54.62
LLaVA-Video-7B (Zhang et al., 2024b)	69.80	59.63	66.38	50.56	72.28	61.41	63.34	51.18	64.19	9.68	41.68	52.51
LongVU-7B (Shen et al., 2024)	55.70	49.54	59.48	48.31	68.32	63.04	57.40	43.10	66.22	9.14	39.49	48.45
Training-based Online MLLMs												
VideoLLM-online-8B (Chen et al., 2024a)	8.05	23.85	12.07	14.04	45.54	21.20	20.79	22.22	18.80	12.18	17.73	19.26
Flash-VStream-7B (Zhang et al., 2025)	25.50	32.11	29.31	33.71	29.70	28.80	29.86	36.36	33.78	5.91	25.35	27.61
Dispider-7B (Qian et al., 2025)	57.72	49.54	62.07	44.94	61.39	51.63	54.55	48.48	55.41	4.30	36.06	45.31
TimeChat-Online-7B (Yao et al., 2025)	75.20	46.80	70.70	47.80	69.30	61.40	61.90	55.90	59.50	9.70	41.70	51.80
StreamBridge-7B (Wang et al., 2025)	84.56	71.56	74.14	49.44	75.25	72.83	71.30	–	–	–	–	–
StreamForest-7B (Zeng et al., 2025)	68.46	53.21	71.55	47.75	65.35	60.87	61.20	58.92	64.86	32.26	52.02	56.61
Streamo-7B (Xia et al., 2026)	79.19	57.80	75.00	49.44	64.36	70.11	65.98	54.55	52.03	31.72	46.10	56.04
ThinkStream-7B (Liu et al., 2026)	87.92	59.63	75.00	53.93	70.30	67.93	69.12	51.17	64.19	66.67	60.68	64.90
Training-free Offline-to-Online Methods												
LLaVA-OV-7B (Li et al., 2024a)	67.79	55.05	72.41	48.31	72.28	62.50	63.06	57.24	55.41	18.28	43.64	53.35
+ ReKV (Di et al., 2025)	52.35	54.13	69.83	43.26	67.33	57.07	57.33	57.58	56.08	18.82	44.16	50.75
+ HERMES (Zhang et al., 2026)	72.48	62.39	74.14	50.56	73.27	65.22	66.34	**60.61**	**61.49**	28.49	50.20	58.27
+ **Ours**	**94.63**	**77.06**	**77.59**	**64.61**	**78.22**	**76.09**	**78.03**	55.89	56.76	**38.17**	**50.27**	**64.15**
Qwen2.5-VL-7B (Bai et al., 2025b)	67.79	55.05	67.24	42.13	66.34	60.87	59.90	51.52	58.78	23.66	44.65	52.28
+ HERMES (Zhang et al., 2026)	85.23	64.22	71.55	53.37	74.26	65.22	68.98	48.48	**62.16**	37.63	49.43	59.21
+ **Ours**	**92.62**	**74.31**	**81.03**	**68.54**	**77.23**	**79.89**	**78.94**	**55.89**	60.14	**43.55**	**53.19**	**66.06**
Qwen3-VL-8B (Bai et al., 2025a)	77.18	60.55	77.59	58.43	70.30	65.22	68.21	53.20	58.11	27.96	46.42	57.32
+ HERMES (Zhang et al., 2026)	87.92	74.31	79.31	56.18	70.30	71.74	73.29	55.56	67.57	24.73	49.28	61.29
+ **Ours**	**95.30**	**85.32**	**83.62**	**65.73**	**79.21**	**81.52**	**81.78**	**56.57**	**68.24**	**44.09**	**56.30**	**69.04**

A.2 OVO-BENCH

OVO-Bench (Li et al., 2025) evaluates online video reasoning and temporal awareness using 644 videos and approximately 2,800 multiple-choice question-answer pairs. Its 12 tasks are grouped into Real-Time Visual Perception, Backward Tracing, and Forward Active Responding. Since we do not focus on the proactive responding ability of MLLMs in this paper, we evaluate on the Real-Time Visual Perception and Backward Tracing subsets, which assess understanding of current visual inputs and previously observed events, respectively. Table 7 reports the full results on these two subsets.

Table 7 shows that ResKV-Tree achieves the highest overall accuracy within each backbone group, surpassing HERMES by 5.88, 6.85, and 7.75 percentage points on LLaVA-OV-7B, Qwen2.5-VL-7B, and Qwen3-VL-8B, respectively. It leads all six Real-Time Visual Perception tasks across the three backbone groups, demonstrating consistent benefits for understanding current visual content. On Qwen3-VL-8B, ResKV-Tree further achieves the best results on all three Backward Tracing tasks within the same backbone group and reaches 69.04% overall accuracy, the highest among all listed methods. These results highlight the value of retaining complementary historical evidence in residual children and retrieving relevant KV pairs according to the question, supporting both real-time perception and reasoning over previously observed events.

B ADDITIONAL IMPLEMENTATION DETAILS

B.1 ADDITIONAL HYPERPARAMETERS

We use binary trees with a merge arity of 2 and maintain at most $N = 8$ historical roots per layer. The merge score weights are $(\lambda_D, \lambda_T, \lambda_S) = (1, 0.1, 0.5)$ for semantic distance, temporal distance, and saliency, respectively, with temporal sensitivity parameter $\tau = 2$. During tree construction, KV-NMS uses a pruning ratio of $\alpha = 0.5$, removing $\lfloor \frac{L}{2} \rfloor$ of the L candidate KV pairs at each merge. We set the parent promotion ratio to $\beta = 0.5$, so the $\lfloor \frac{L'}{2} \rfloor$ most salient pairs among the L' retained KV pairs are promoted to the parent, while the others remain in the residual children. Following HERMES's 4K budget setting, we set the historical retrieval budget to $B = 4K$ KV pairs per layer.

Table 8: Ablation on tree arity with Qwen2.5-VL-7B. The baseline denotes Qwen2.5-VL-7B without ResKV-Tree.

Method	StreamingBench	OVO-Bench	MLVU
baseline	73.3	52.3	66.9
$n = 2$	**80.5**	**66.1**	**69.1**
$n = 3$	80.1	65.2	68.4
$n = 4$	79.8	65.0	68.5

To evaluate the default merge arity, Table 8 compares binary, ternary, and quaternary merging. Binary merging achieves the highest accuracy on all three benchmarks.

B.2 Cross-layer KV Cache Length Alignment

Each layer searches its residual KV trees independently and collects complete nodes until the historical candidate length reaches the stopping threshold B or the queue is exhausted. The complete final node can take the candidate length beyond B, and candidate lengths can differ across layers. Before cache assembly, we align these lengths to a common target. Let H_ℓ denote the historical candidate length returned by layer ℓ. We set

$$H^\star = \min\left(B, \min_\ell H_\ell\right). \tag{5}$$

We first stably order each layer's historical candidates by their source temporal coordinates, then apply KV-NMS using the current question's saliency and target $L' = H^\star$. Layers already at this target retain their candidates in full. This implementation step aligns the historical cache lengths across layers within the answering budget while preserving the stored trees.

C Algorithms

Algorithms 1, 2, and 3 detail residual KV tree construction, KV-NMS, and question-guided retrieval, respectively. We use the notation of Section 3. Each node stores its current KV pairs, their source coordinates and pseudo-question saliency scores, its frame interval, and references to its children. All subset and concatenation operations preserve these attributes. Construction is performed independently for each Transformer layer, and KV-NMS is shared by construction and retrieval-time length alignment.

Algorithm 1 Residual KV Tree Construction

Require: Layer forest $\mathcal{T}_\ell$, new historical leaf u, root limit N, pruning ratio α, promotion ratio β
Ensure: Updated residual KV forest $\mathcal{T}_\ell$

Insert u into the roots of $\mathcal{T}_\ell$
while $|\,\mathrm{Roots}(\mathcal{T}_\ell)| > N$ **do**
  Compute root descriptors, mean saliency, and normalized temporal midpoints
  $(n, m) \leftarrow \arg\min_{n \neq m} J(n, m)$ over current root pairs using Equation 2
  $\mathcal{C} \leftarrow \mathrm{Concat}(\mathrm{KV}(n), \mathrm{KV}(m))$
  $L \leftarrow |\mathcal{C}|$, $L' \leftarrow L - \lfloor \alpha L \rfloor$
  $\mathcal{I}_n, \mathcal{I}_m \leftarrow$ indices in $\mathcal{C}$ originating from n and m
  $\mathcal{I}_r \leftarrow$ KV-NMS$(\mathcal{C}, s^{\mathrm{pseudo}}, L')$ ▷ Algorithm 2
  $p \leftarrow \lfloor \beta L' \rfloor$
  $\mathcal{P} \leftarrow$ indices of the p highest-saliency KV pairs in $\mathcal{I}_r$
  $\mathcal{R}_n \leftarrow (\mathcal{I}_r \cap \mathcal{I}_n) \setminus \mathcal{P}$
  $\mathcal{R}_m \leftarrow (\mathcal{I}_r \cap \mathcal{I}_m) \setminus \mathcal{P}$
  Replace the KV pairs of n and m with $\mathcal{C}[\mathcal{R}_n]$ and $\mathcal{C}[\mathcal{R}_m]$
  Preserve the existing descendants of n and m
  Create parent v with KV pairs $\mathcal{C}[\mathcal{P}]$ and children (n, m)
  $[a_v, b_v) \leftarrow [\min(a_n, a_m), \max(b_n, b_m))$
  $\mathrm{level}(v) \leftarrow 1 + \max(\mathrm{level}(n), \mathrm{level}(m))$
  Replace roots n, m with v in $\mathcal{T}_\ell$
end while
return $\mathcal{T}_\ell$

Algorithm 2 KV Non-Maximum Suppression

Require: KV pairs $\mathcal{C}$, saliency scores s, target count M with $0 \le M \le |\mathcal{C}|$
Ensure: Retained index set $\mathcal{I}$ in the original input order

function KV-NMS($\mathcal{C}, s, M$)
  $L \leftarrow |\mathcal{C}|$
  if $M = 0$ **then**
    return $\emptyset$
  end if
  $\pi \leftarrow$ stable descending sort of indices by s
  $\mathbf{v}_i \leftarrow$ values of KV pair i concatenated across KV heads
  $d_1 \leftarrow -\infty$
  for $r = 2, \ldots, L$ **do**
    $d_r \leftarrow \max_{u<r} \cos(\mathbf{v}_{\pi(r)}, \mathbf{v}_{\pi(u)})$
  end for
  $\mathcal{D} \leftarrow$ the $L - M$ ranks with largest d_r, breaking ties by larger rank r
  $\mathcal{I} \leftarrow \{\pi(r) : r \notin \mathcal{D}\}$, sorted by original input index
  return $\mathcal{I}$
end function

Algorithm 3 Question-Guided Retrieval

Require: Layer forests $\{\mathcal{T}_\ell\}$, user question x, historical KV budget $B > 0$
Ensure: Retrieved historical KV pairs $\{\mathcal{H}_\ell\}$ with a common length at most B

Compute layerwise pre-RoPE question queries $\{\bar{\mathbf{q}}_{\ell,h}\}$ from x
for each Transformer layer ℓ **do**
  $\mathcal{Q} \leftarrow$ empty max-priority queue, $\mathcal{H}_\ell \leftarrow [\,]$, $c \leftarrow 0$
  for each root n of $\mathcal{T}_\ell$ **do**
    Score and enqueue n using NODESCORE(n, x, ℓ)
  end for
  while $\mathcal{Q} \neq \emptyset$ and $c < B$ **do**
    $n \leftarrow \mathrm{PopMax}(\mathcal{Q})$
    Append all current KV pairs of n and their question saliency to $\mathcal{H}_\ell$
    $c \leftarrow c + |\,\mathrm{KV}(n)|$
    if $c \ge B$ **then**
      break
    end if
    for each child v of n **do**
      Score and enqueue v using NODESCORE(v, x, ℓ)
    end for
  end while
  Stably sort $\mathcal{H}_\ell$ by source temporal coordinates
end for
$H^\star \leftarrow \min(B, \min_\ell |\mathcal{H}_\ell|)$
for each Transformer layer ℓ **do**
  $\mathcal{I}_\ell \leftarrow$ KV-NMS($\mathcal{H}_\ell, s^x_\ell, H^\star$)
  $\mathcal{H}_\ell \leftarrow \mathcal{H}_\ell[\mathcal{I}_\ell]$
end for
return $\{\mathcal{H}_\ell\}$
function NODESCORE(n, x, ℓ)
  if $|\,\mathrm{KV}(n)| = 0$ **then**
    return $+\infty$
  end if
  Compute and cache question saliency $s^x_{\ell,i}$ for $i \in \mathcal{I}_n$ using Equation 1
  return $|\mathcal{I}_n|^{-1} \sum_{i \in \mathcal{I}_n} s^x_{\ell,i}$
end function

Algorithm 3 retains the complete final node even when its inclusion exceeds B, then applies the common target in Equation 5. Empty nodes receive the highest priority so that their residual descendants remain accessible. Queue ties are resolved by frame interval and node identifier. Retrieval and length alignment operate on temporary copies of the selected KV pairs and preserve the stored forests.

D ADDITIONAL VISUALIZATIONS

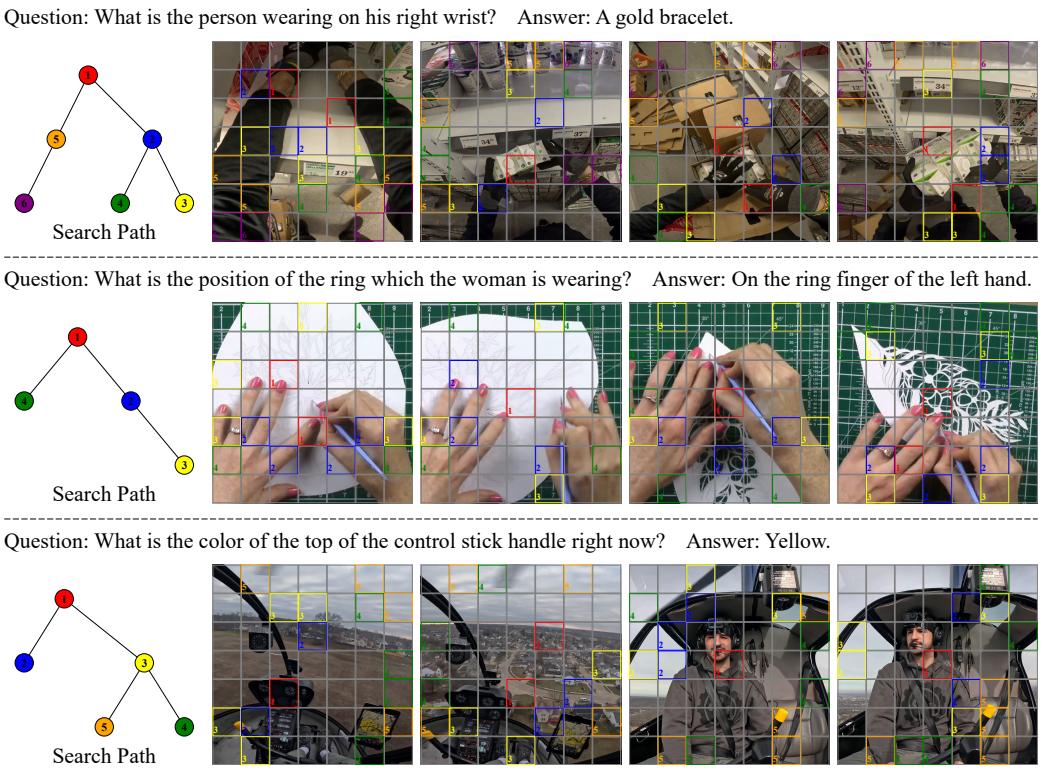

Figure 6: Additional visualization examples of question-guided retrieval in ResKV-Tree for questions about a bracelet (top), a ring (middle), and a control-stick handle (bottom). Each example shows four randomly sampled frames and the retrieval path through one tree in a randomly selected Transformer layer. Each frame uses a 7×7 grid, with patch labels identifying the tree nodes holding KV pairs from those locations.

Figure 6 provides further examples of retrieving small visual details together with their surrounding context, using the same frame sampling, layer selection, and patch annotation procedure as Section 4.4. In the bracelet example, the retrieved regions include the hands and wrists during interactions with store products, providing evidence for identifying the accessory on the right wrist. In the ring example, regions associated with residual node 2 cover the fingers and ring, complementing the root regions around the ongoing craft activity. In the control-stick example, residual node 5 includes the yellow handle top, while other nodes retain regions of the cockpit, pilot, and exterior scene. The target details thus occur at different locations within the retrieved trees. These examples illustrate why retaining complementary KV pairs in residual children is useful: question-relevant attributes remain individually accessible alongside the salient evidence stored in parents.