StreamMeCo: Long-Term Agent Memory Compression for Efficient Streaming Video Understanding

Junxi Wang[1,2], Te Sun[1], Jiayi Zhu[1], Junxian Li[1]
Haowen Xu[1], Zichen Wen[1,3], Xuming Hu[4], Zhiyu Li[5], Linfeng Zhang[1]†
[1]Shanghai Jiao Tong University, [2]Fudan University
[3]Shanghai AI Laboratory, [4]Hong Kong University of Science and Technology
[5]MemTensor (Shanghai) Technology Co., Ltd.
junxiwang182@gmail.com, zhanglinfeng@sjtu.edu.cn

Abstract

Vision agent memory has shown remarkable effectiveness in streaming video understanding. However, storing such memory for videos incurs substantial memory overhead, leading to high costs in both storage and computation. To address this issue, we propose **StreamMeCo**, an efficient **Stream** Agent **Me**mory **Co**mpression framework. Specifically, based on the connectivity of the memory graph, StreamMeCo introduces edge-free minmax sampling for the isolated nodes and an edge-aware weight pruning for connected nodes, evicting the redundant memory nodes while maintaining the accuracy. In addition, we introduce a time-decay memory retrieval mechanism to further eliminate the performance degradation caused by memory compression. Extensive experiments on three challenging benchmark datasets (M3-Bench-robot, M3-Bench-web and Video-MME-Long) demonstrate that under **70%** memory graph compression, StreamMeCo achieves a **1.87**× speedup in memory retrieval while delivering an average accuracy improvement of **1.0%**. *Our code is available at https://github.com/Celina-love-sweet/StreamMeCo.*

1 Introduction

With the rapidly growing demands of application scenarios such as live streaming (Lu et al., 2018), real-time surveillance (Rao et al., 2022), and autonomous driving (Yurtsever et al., 2020), research on streaming video understanding has become increasingly important. Unlike offline video understanding (Zuo et al., 2025; Tang et al., 2025; Kahatapitiya et al., 2025; Wang et al., 2025b), where models can access the complete video content and user questions prior to inference, streaming video understanding (Wang et al., 2025a; Di et al., 2025; Xiong et al., 2025; Qian et al., 2025; Zhang et al., 2024; Yang et al., 2025a) can only rely on the limited information observed before the user's question arrives, making how to efficiently process continuously incoming visual information a critical challenge. Current research on streaming video understanding primarily focuses on achieving long-video understanding by carefully handling the vision tokens and KV cache (Yang et al., 2025c; Yao et al., 2025a). However, for extremely long videos, these methods still suffer from the loss of critical visual information and struggle to maintain long-term consistency of entities such as human identities.

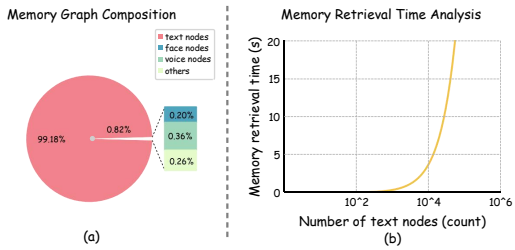


Figure 1: An example from M3-Bench-robot. (a) Composition of the memory graph. (b) Analysis of memory retrieval time with respect to the number of text nodes, indicating that the *increment of memory nodes makes the retrieval time not acceptable.*

To tackle this challenge, agent memory-based methods are introduced to organize the information of videos as memory in plaintext. For example, M3-Agent (Long et al., 2025) leverages agent memory mechanism to model streaming video as a memory graph, representing entities with face and voice nodes and abstracting rich visual content into text nodes, thereby preserving information integrity and long-term entity consistency. Nevertheless, as the memory graph scales up, storage and retrieval efficiency become major bottlenecks. As shown in Figure 1, the increasing number of text nodes leads to a rapidly increasing memory retrieval latency, which hinders the real-time answering of questions.

†Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 13234–13251

To address this, we propose ***StreamMeCo***, an efficient training-free Long-Term ***Stream*** *Agent* ***Me****mory* ***Co****mpression* framework, achieving the first memory compression of industrial-level streaming video agent and supporting efficient memory graph retrieval. Based on the connectivity of the memory graph, we design a dual-branch memory compression method. For isolated text nodes, we use the ***Edge-free Minmax sampling (EMsampling)*** module for compression. First, we cluster the text nodes into different groups, then for each cluster, we first select the node closest to the cluster center and add it to the selected set, then calculate the minimum distance between each unselected node and the selected set, choosing the farthest point. This process continues until the number of selected nodes meets the predefined requirement. For connected text nodes, we use the ***Edge-aware Weighting pruning (EW-pruning)*** module. By calculating the weight edge matrix between text nodes and face nodes or voice nodes, we determine the entity importance of each text node. We then combine the embedding similarity of the text nodes to select the most dissimilar text nodes, and ultimately determine the retained text nodes based on a comprehensive fusion score.

Additionally, to reduce the performance degradation caused by compressing the memory graph, inspired by the human memory mechanism, we design a ***Time-decay Memory Retrieval (TMR)*** mechanism for efficiently retrieving the memory graph. We calculate the total similarity between text nodes in the memory graph and the model's query for each time segment. Based on the total similarity of each time segment, we dynamically allocate the number of text nodes to different segments. Furthermore, to simulate the memory decay mechanism of the human brain, we assign nonlinear time-varying decay to the text nodes of each time segment, thereby adaptively balancing long-term memory and recent critical information during the retrieval stage, and guaranteeing the retrieval of more memories from recent segments.

In summary, the main contributions of this paper are as follows:

- We propose *edge-free minmax sampling* module and *edge-aware weighting pruning* module, which efficiently compress the memory graph based on its structural properties.
- We introduce a novel *time-decay memory retrieval* mechanism, which has been experimentally proven to significantly outperform previous retrieval frameworks.
- We demonstrate the effectiveness of *StreamMeCo* through extensive experiments. On three high-difficulty benchmark datasets, under approximately **70%** memory graph compression, the proposed method achieves a **1.87**$\times$ speedup in memory retrieval while delivering an average accuracy improvement of **1.0%**.

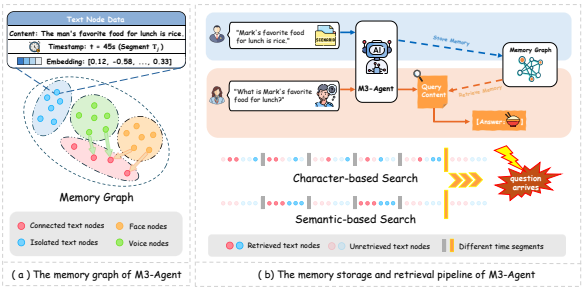
2 Related Work

2.1 Streaming Video Understanding

Streaming video understanding enables a model to continuously process incoming frames, and upon receiving a user question Q_t at time t, generate responses based on the accumulated content $C_{1:t}$. Existing methods fall into two main paradigms: visual token compression and KV cache compression. The former selects informative tokens either in pixel space or after feature encoding (Chen et al., 2025; Dorovatas et al., 2025; Zeng et al., 2025; Yao et al., 2025a; Li et al., 2025b; Huang et al., 2025b; Chen et al., 2024). While the latter maintains a short-term sliding window and a long-term memory stack at the LLM level to preserve information across temporal scales (Yang et al., 2025c; Kim et al., 2025; Ning et al., 2025; Di et al., 2025; Xu et al., 2025b). Recently, M3-Agent (Long et al., 2025) departs from these designs by integrating long-term agent memory into streaming video understanding, showing strong potential for modeling long-range temporal dependencies and maintaining entity consistency.

2.2 Long-Term Agent Memory

Long-term agent memory is designed to provide LLM-based agents with persistent memory beyond a single context window. Depending on the type of input it handles, it can be categorized into unimodal and multimodal forms. Unimodal memory (Yu et al., 2025a; Zhou et al., 2025; Chhikara et al., 2025a; Packer et al., 2023) typically maintains consistency across multi-turn reasoning by constructing structured representations, latent embeddings, or summaries. In contrast, multimodal memory (Yu et al., 2025b; Zhang et al., 2025; Yao et al., 2025b; Fan et al., 2025) integrates visual, linguistic, and other signals into a unified space, updating them to support stable long-term reasoning and decision-making. Building on this foundation, M3-Agent is the first to introduce long-



(a) The memory graph of M3-Agent (b) The memory storage and retrieval pipeline of M3-Agent

Figure 2: The overview of M3-Agent. (a) The memory graph of M3-Agent. (b) The memory storage and retrieval pipeline of M3-Agent, illustrating two different retrieval strategies.

term agent memory into streaming video scenarios, enabling application in real-time settings.

2.3 LLMs and Agent Systems

With the development of LLMs and MLLMs, they have demonstrated strong performance across a variety of core tasks, including retrieval and question answering (Zeng et al., 2026; Zhang et al., 2026b), audio and video understanding (Diao et al., 2025b,a, 2024), temporal modeling (Diao et al., 2025c), and complex reasoning (Yang et al., 2025b; Ma et al., 2026; Madani et al., 2026). Meanwhile, a range of research directions aimed at enhancing their capabilities have also rapidly emerged, such as VLA modeling (Fang et al., 2025), unified multimodal modeling (Han et al., 2026), inference acceleration (Shen et al., 2026; Wang, 2026), and chain-of-thought reasoning optimization (Zhang et al., 2026c; Li et al., 2025a; Zhang et al., 2026a). In addition, LLMs and MLLMs have been widely applied in agent frameworks, covering tool use and policy optimization (Huang et al., 2025a; Yang et al., 2026b,a), multi-agent collaborative planning (Wang et al., 2026), and agentic reinforcement learning (Wu et al., 2026), significantly improving their ability to handle complex real-world scenarios. In the domain of agent memory, a variety of memory frameworks built upon LLMs and MLLMs, such as Mem0 (Chhikara et al., 2025b), MemOS (Li et al., 2025c), and M3-Agent (Long et al., 2025), have shown great potential in long-term information storage and retrieval, further highlighting the central role of large models in intelligent agent systems.

3 Preliminaries

Graph-based agent memory has demonstrated great potential across a variety of research areas. Our method exhibits strong generality and can be readily adapted to different graph-based agent memory frameworks, while this paper focuses on its application in streaming video understanding scenarios. Next, we provide a detailed introduction to M3-Agent (Long et al., 2025).

3.1 Memory Graph of M3-Agent

As shown in Figure 2(a), the memory graph includes face nodes, voice nodes, text nodes, and other relevant information such as edge weights. Each text node contains content, embedding vectors, and a timestamp of the content. Among these nodes, only some voice nodes are connected to text nodes, and some face nodes are connected to text nodes, while there are no internal connections within face nodes, voice nodes, or text nodes, nor between voice and face nodes. Based on the graph's connectivity, the content of isolated text nodes is similar to: "*The man's favorite food for lunch is rice.*" with no specific entity references. In contrast, the content of connected text nodes is like: "*Mark's favorite food for lunch is rice.*" which explicitly includes entity information.

3.2 Retrieval of Memory Graph

As shown in Figure 2(b), after a question is input into M3-Agent, the model performs reasoning and outputs the query content to be retrieved. Based on this content, the model retrieves from the memory graph, which is primarily divided into two approaches: ***character-based*** and ***semantic-based***. If the retrieval content contains characters like "*character id*", the model returns the k most similar content based on the similarity between the embedding of the query content and the content embedding of the text nodes, as the retrieved memory. If there is no "*character id*" identifier, semantic retrieval is performed. In this case, the model segments all text nodes based on their timestamps, selecting the most similar node as the representative for each time segment. Finally, the two most similar text nodes are selected, and all text node contents at the timestamps of these two nodes are returned as the retrieved memory.

4 Methodology

4.1 Overview

The overview of StreamMeCo is illustrated in Figure 3. The framework consists of two components, achieving efficient compression of memory

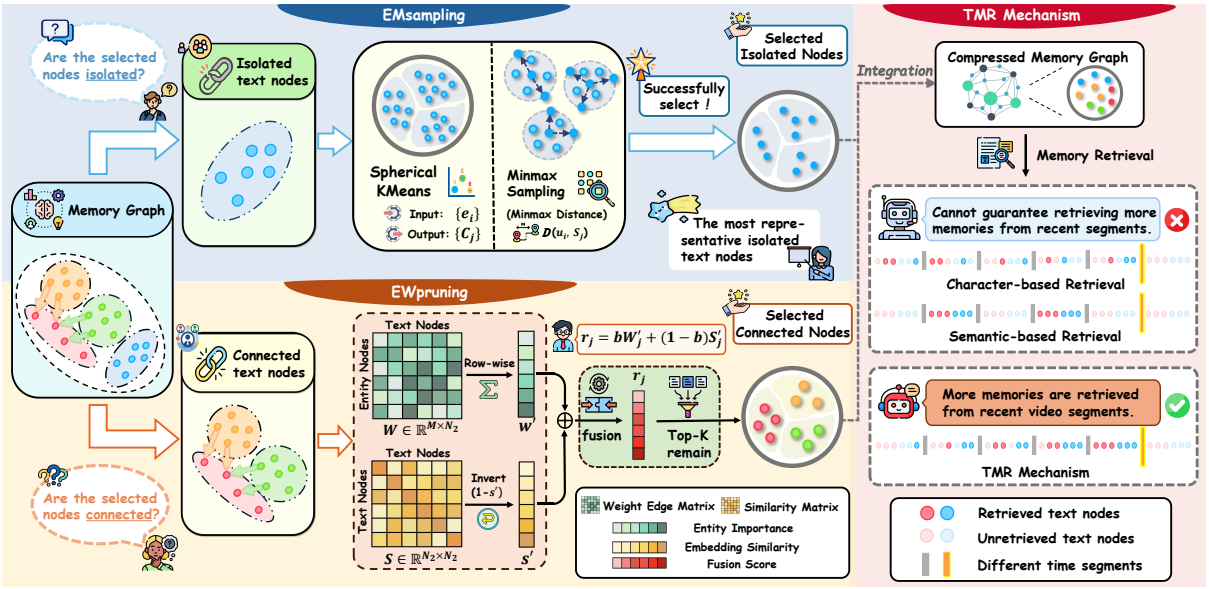


Figure 3: The overview of StreamMeCo. Efficient memory graph compression is achieved via the EMsampling and EWpruning modules, while the TMR mechanism prioritizes retrieving memories from more recent segments, enabling fast and accurate memory retrieval.

graph while supporting fast and accurate retrieval.

4.2 Text Memory Compression

Based on the connectivity of the memory graph, we categorize text nodes into isolated nodes and connected nodes, and accordingly propose a dual-branch memory compression method. Different compression strategies are applied to these two types of text nodes.

Edge-free Minmax Sampling. For isolated text nodes, we have carefully designed the EMsampling module to select the most representative text nodes. Specifically, given N_1 text nodes, we use the *Spherical KMeans Algorithm* (Dhillon and Modha, 2001) to cluster them into $N_1 \times a$ clusters based on the embeddings of the text node contents, as shown in the following formula:

$$\{C_j\} = KMeans(\{e_i\}, a), \tag{1}$$

where e_i $(i = 1, 2, \ldots, N_1)$ represents the embedding vector of the text node contents, and clustering ratio a is used to control the number of clusters, while C_j $(j = 1, 2, \ldots, N_1 \times a)$ represents the j-th cluster.

Then, we assume the retention ratio of isolated text nodes is α, and distribute them evenly across each cluster. The number of text nodes retained in the j-th cluster is $|C_j| \times \alpha$, where $|C_j|$ represents the number of text nodes in the j-th cluster.

Finally, we adopt an effective *minmax sampling* strategy to select the text nodes to be retained from each cluster. Specifically, for the j-th cluster, we first select the text node closest to the cluster center and add it to the selected node set S_j. Then, we compute the distance from each unselected text node in the unselected node set U_j to the selected node set S_j, and select the node with the largest distance to add to S_j. The distance from each unselected node to the selected node set S_j is defined as the minimum distance to any node in the selected set, as shown in the following formula:

$$D(u_i, S_j) = \min_{s_k \in S_j} \|e_{u_i} - e_{s_k}\|, \tag{2}$$

where $u_i \in U_j$ is an unselected node, and $s_k \in S_j$ is a selected node, while e_{u_i} and e_{s_k} represent the embedding vectors of the unselected and selected nodes, respectively.

We select unselected node u_{i^*} with the farthest distance and add it to the selected node set S_j:

$$u_{i^*} = \arg\max_{u_i \in U_j} D(u_i, S_j). \tag{3}$$

Until the number of selected nodes reaches $|C_j| \times \alpha$, the number of retained text nodes.

Edge-aware Weighting Pruning. For connected text nodes, we carefully design the EWpruning module, thereby constructing a more compact yet informative memory representation to enable efficient node pruning. Specifically, given N_2 text nodes that are connected to face and voice nodes via edges, we construct an entity-based weight edge matrix $W \in \mathbb{R}^{M \times N_2}$, where

M denotes the total number of face and voice nodes. Each entry w_{ij} represents the edge weight between the i-th face or voice node and the j-th text node, reflecting the importance of the j-th text node to the i-th face or voice node.

For the weight matrix W, the row-wise summation is computed as $W' = \sum_{i=1}^{M} w_{ij}$, where $W' \in \mathbb{R}^{1\times N_2}$ represents the importance scores of the N_2 connected text nodes with respect to the set of face and voice entity nodes. Subsequently, we normalize W' to the range $[0, 1]$ to obtain the final *entity importance* for each text node.

Next, for the N_2 text nodes, we compute their pairwise similarities to obtain a similarity matrix $S \in \mathbb{R}^{N_2\times N_2}$, where s_{ij} denotes the embedding similarity between the i-th and j-th text nodes. We then perform a row-wise summation on the similarity matrix, computed as $S' = \sum_{i=1}^{N_2} s_{ij}$, where $S' \in \mathbb{R}^{1\times N_2}$ represents the total similarity of each text node to all other text nodes. Since we aim to select the text nodes with higher diversity, we normalize S' to the range $[0, 1]$ and then invert it as $1 - S'$, resulting in the *embedding similarity* for each text node relative to the connected text node set. For brevity, we continue to denote the processed entity importance and embedding similarity as W' and S'.

We assume the retention ratio for connected text nodes is β, meaning that we need to keep a total of $N_2 \times \beta$ text nodes. For the j-th connected text node, we compute its normalized entity importance W'_j and its embedding similarity S'_j through a weighted combination. The final fusion score is computed as:

$$r_j = bW'_j + (1 - b)S'_j, \tag{4}$$

where the balancing coefficient b controls the trade-off between entity importance and embedding similarity.

Finally, we retain the top $N_2 \times \beta$ text nodes with the highest scores and prune the others.

4.3 Time-decay Memory Retrieval

To reduce the performance degradation caused by compressing the memory graph, inspired by the human memory mechanism, we propose a TMR framework. Specifically, given the query content generated by the model in Section 3.2, we first compute the embedding similarity between the query and the content embeddings of all text nodes. According to their timestamps, these text nodes are grouped into a series of chronologically ordered time segments T_j ($j = 1, 2, \ldots, t$). At time step t, when the system receives a query, the relevance score of each time segment is computed by aggregating the similarity scores of all text nodes within that segment as $E_j = \sum_{i=1}^{|T_j|} s_{ij}$, where s_{ij} denotes the embedding similarity between the query and the i-th text node in T_j, E_j denotes the aggregated similarity within the time segment T_j, and s_{ij} denotes the embedding similarity between the query content and the i-th text node in time segment T_j, and $|T_j|$ represents the number of text nodes in segment T_j.

Next, to simulate human memory decay, we first compute the average similarity of each time segment as $\bar{E}_j = \frac{E_j}{|T_j|}$. Based on this averaged relevance score, we then apply a temporal decay function to model the fading of older memories:

$$E'_j = \bar{E}_j \cdot e^{-\lambda(t-j)}, \tag{5}$$

where $j = 1, 2, \ldots, t$ indicates the temporal position of the segment, and the memory decay coefficient λ controls the rate of memory decay.

To be consistent with the number of retrieved text nodes in Section 3.2, we assume that the model needs to retrieve a total of k text nodes as the final memory output. Accordingly, we compute the number of nodes selected from each time segment as:

$$Num_j = \frac{E'_j}{\sum_{i=1}^{t} E'_i} \times k, \tag{6}$$

where Num_j denotes the number of nodes selected from the j-th segment.

Finally, within each time segment, we select the top-ranked text nodes according to their similarity to the query content until reaching Num_j nodes, which constitute the final retrieved memory.

5 Experiments

5.1 Experiments Setup

Datasets. We select three challenging open-source benchmark datasets, M3-Bench-robot, M3-Bench-web (Long et al., 2025) and Video-MME-Long (Fu et al., 2025), to evaluate the effectiveness of StreamMeCo. M3-Bench-robot is a streaming video dataset, while the others are offline video datasets. Consistent with the original M3-Agent setup, we use GPT-4o to assess the answer quality and employ text-embedding-3-large

to encode the query content generated by M3-Agent to supporting subsequent memory graph retrieval. For more details, please refer to the Appendix A.1.

Baselines. Our baselines include closed-source MLLMs: Gemini-1.5-Pro (Team et al., 2024), GPT-4o (Hurst et al., 2024), Gemini-Agent and Gemini-GPT4o-Hybrid (Long et al., 2025). open-source MLLMs: Qwen2.5-VL-7B-Instruct (Bai et al., 2025), Qwen2.5-Omni-7B (Xu et al., 2025a). As well as streaming video models: MovieChat (Song et al., 2024), MA-LMM (He et al., 2024), VideoChat-Online (Huang et al., 2025b), Flash-VStream (Zhang et al., 2024), TimeChat-Online (Yao et al., 2025a), StreamForest (Zeng et al., 2025), and M3-Agent (Long et al., 2025). For more introductions about the baselines, please refer to the Appendix A.2.

Implementation Details. All experiments are conducted on two NVIDIA A100 (80G) GPUs. To reduce error, each experiment is executed three times and the average result is reported. In our setup, the clustering ratio a is set to 0.05, the balancing coefficient b is set to 0.1, and the memory decay coefficient λ is set to 0.1. For the M3-Bench-robot and M3-Bench-web, we use the memory graphs provided by the M3-Agent paper, for Video-MME-Long, we replace the Gemini-1.5-Pro API with Gemini-2.5-Pro, while keeping all other settings consistent with M3-Agent. We apply compression only to the memory graph produced by M3-Agent, while keeping all other components identical to the original model.

5.2 Main Results

To comprehensively evaluate the performance of our method, we conduct comparisons with 13 competitive baselines. For more details about the baselines, please refer to the Appendix A.3. As shown in Table 1, the overall performance of **StreamMeCo** significantly surpasses that of existing closed-source and open-source MLLMs, as well as other streaming video models, demonstrating the effectiveness of the agent memory mechanism for streaming video understanding. Even with **70%** of text nodes compressed, StreamMeCo and TMR mechanism still achieves an average accuracy improvement of **1.0%** across all datasets compared to the uncompressed M3-Agent.

In addition, we have constructed five memory graph compression methods and compared them with StreamMeCo, with the compression ratio controlled at 30%. In the random method, we randomly selected nodes from two types of text nodes for compression. For the clustering method, we used the KMeans clustering algorithm to evaluate the differences between our method and traditional clustering approaches. Specifically, we divided the two types of text nodes into a series of smaller clusters, each of which retains the text node closest to the cluster center. For the DART method, we were inspired by the DART (Wen et al., 2025). Specifically, for the two types of text nodes, we first randomly selected 2% of the pivot nodes, then calculated the total similarity of the remaining nodes with these pivot nodes, prioritizing the compression of nodes with higher similarity to the pivot nodes, until the target compression ratio was reached. Finally, we also draw inspiration from the streaming video understanding model TimeChat-Online (Yao et al., 2025a) and the LLM memory method MemoryLLM (Wang et al., 2024), which we denote as TimeChat-Memory and MemoryLLM, respectively. For TimeChat-Memory, based on the chronological order of Agent Memory, we compute the similarity between two adjacent memory nodes. If the similarity exceeds 0.7, the earlier memory node is removed; otherwise, both are retained. Meanwhile, following the original design, the memory buffer adopts a first-in-first-out (FIFO) strategy, once the target compression ratio is exceeded, earlier memories are preferentially discarded. For MemoryLLM, a fixed-capacity memory bank is maintained, where memories are continuously written in chronological order. When the memory bank reaches its capacity, each newly incoming memory is compared with the existing ones, and the most similar old memory is removed to make room for the new memory. As shown in the Table 1, StreamMeCo significantly outperforms these methods in terms of performance.

For the M3-Bench-robot dataset, all three compression ratios lead to varying degrees of performance improvement. In contrast, the performance gains on the M3-Bench-web and Video-MME-Long datasets are relatively limited. This is because M3-Bench-robot consists of real-world filming captured from a first-person robotic perspective, involving daily environments and human–robot interactions, which naturally contain a higher level of redundancy. In comparison,

Dataset	M3-Bench-robot						M3-Bench-web						Video-MME-Long	Avg.
Model	ME	MH	CM	PU	GK	**All**	ME	MH	CM	PU	GK	**All**		
Closed-source MLLMs														
Gemini-1.5-Pro	6.5	7.5	8.0	9.7	7.6	8.0	18.0	17.9	23.8	23.1	28.7	23.2	38.0	23.1
GPT-4o	9.3	9.0	8.4	10.2	7.3	8.5	21.3	21.9	30.9	27.1	39.6	28.7	38.8	25.3
Gemini-Agent	15.8	17.1	15.3	20.0	15.5	16.9	29.3	20.9	33.8	34.6	45.0	34.1	55.1	35.4
Gemini-GPT4o-Hybrid	21.3	25.5	22.7	28.8	23.1	24.0	35.9	26.2	37.6	43.8	52.2	41.2	56.5	40.6
Open-source MLLMs														
Qwen2.5-VL-7b-Instruct	2.9	3.8	3.6	4.6	3.4	3.4	11.9	10.5	13.4	14.0	20.9	14.9	46.9	21.7
Qwen2.5-Omni-7b	2.1	1.4	1.5	1.5	2.1	2.0	8.9	8.8	13.7	10.8	14.1	11.3	42.2	18.5
Streaming Video Models														
MovieChat [CVPR24]	13.3	9.8	12.2	15.7	7.0	11.2	12.2	6.6	12.5	17.4	11.1	12.6	19.4	14.4
MA-LMM [CVPR24]	25.6	23.4	22.7	39.1	14.4	24.4	26.8	10.5	22.4	39.3	15.8	24.3	17.3	22.0
VideoChat-Online [CVPR25]	31.7	24.7	30.5	43.1	17.1	29.9	34.3	18.9	34.1	48.3	23.1	32.7	45.9	36.2
Flash-VStream [ICCV25]	21.6	19.4	19.3	24.3	14.1	19.4	24.5	10.3	24.6	32.5	20.2	23.6	25.0	22.7
TimeChat-Online [MM25]	35.6	29.4	31.5	44.3	22.3	33.6	38.8	22.8	38.4	51.6	28.0	36.6	49.2	39.8
StreamForest [NIPS25]	33.4	29.4	31.3	44.3	19.0	32.1	37.9	22.3	39.9	51.7	27.6	36.5	51.9	40.2
M3-Agent [ICLR26]	30.9	29.4	29.6	41.1	22.6	30.3	44.9	25.6	44.8	58.6	53.7	47.9	**56.0**	44.7
+Random (↓ 30%)	28.9	29.4	27.7	40.5	18.0	28.5	42.4	24.7	40.1	54.8	50.8	44.6	54.2	42.4
+Clustering (↓ 30%)	30.0	32.9	30.0	42.5	20.8	29.6	41.2	26.3	41.7	55.9	50.9	45.3	54.6	43.2
+DART (↓ 30%)	29.5	24.7	25.8	40.1	21.7	29.1	40.4	26.0	41.3	55.6	51.2	45.4	54.8	43.1
+TimeChat-Memory (↓ 30%)	31.8	36.5	31.7	42.2	18.7	30.2	39.8	27.1	38.9	56.6	46.7	44.7	55.0	43.3
+MemoryLLM (↓ 30%)	31.4	34.1	30.9	41.4	18.0	28.9	38.4	24.9	36.6	56.5	45.4	44.5	53.9	42.8
+StreamMeCo (↓ 30%)	32.9	30.6	30.9	42.5	19.6	30.7	41.3	26.0	44.3	58.3	50.5	47.0	54.8	44.2
+StreamMeCo (↓ 50%)	32.3	28.2	29.8	41.4	21.1	30.6	39.7	25.2	38.9	58.4	47.3	44.7	54.4	43.2
+StreamMeCo (↓ 70%)	29.1	30.6	26.9	40.3	19.9	28.4	37.0	21.7	36.1	53.0	46.3	41.9	53.3	41.2
+StreamMeCo+TMR (↓ 30%)	35.7	32.9	35.5	44.0	27.5	**34.6**	46.4	32.2	46.5	61.8	55.8	**50.7**	**55.2**	**46.8**
+StreamMeCo+TMR (↓ 50%)	34.8	34.1	32.4	43.1	24.5	**33.4**	45.4	31.3	45.0	59.2	52.7	**48.9**	**56.6**	**46.3**
+StreamMeCo+TMR (↓ 70%)	34.9	31.8	32.1	42.7	24.8	**34.2**	43.8	28.9	43.9	57.9	52.9	**47.9**	54.9	**45.7**

Table 1: Performance comparison of different models on M3-Bench-robot, M3-Bench-web and Video-MME-Long. The best three results for the streaming video models are highlighted in **bold black font**.

both M3-Bench-web and Video-MME-Long are sourced from YouTube, where videos typically exhibit stronger narrative structures or scripted production, resulting in more concise content and relatively less redundancy, leaving limited room for effective compression.

5.3 Ablation Study

Due to the increased overlap between the nodes compressed by StreamMeCo and those retained through random compression at higher text-node compression ratios, the performance differences between modules become less distinguishable. To highlight the contribution of each component in our compression, we conduct the components ablation analysis under a 30% compression ratios.

Compression Components Ablation Analysis. This component analysis focuses solely on the compression method and does not involve the TMR mechanism. As shown in Table 2, we evaluate the performance of EMsampling (A), entity importance (B), and embedding similarity (C)

Components			M3-Bench-Robot	M3-Bench-Web
A	**B**	**C**	Accuracy	Accuracy
–	–	–	30.3 (100.0%)	47.9 (100.0%)
			28.5 (94.1%)	44.6 (93.1%)
✓			30.3 (100.0%)	45.7 (95.4%)
	✓	✓	30.1 (99.3%)	46.1 (96.2%)
✓	✓		30.2 (99.7%)	46.1 (96.2%)
✓		✓	30.5 (100.7%)	46.6 (97.3%)
✓	✓	✓	**30.7 (101.3%)**	**47.0 (98.1%)**

Table 2: Ablation study of the compression components. The first row reports the accuracy of the M3-Agent without compression. **A**: EMsampling; **B**: entity importance; **C**: embedding similarity.

under different combinations. The results indicate that enabling any single component leads to varying degrees of performance improvement. Specifically, the EMsampling module effectively selects the most representative text nodes from the isolated-node subset, while for connected text nodes, our method jointly considers both entity importance and embedding similarity, yielding a more compact yet informative memory representation. When all three components are used together, the model achieves its best performance on both benchmark datasets, reaching accuracies of **30.7%** and **47.0%**, corresponding to **101.3%** and **98.1%** of the original M3-Agent, respectively. In contrast, random compression achieves only 28.5% and 44.6% accuracy on the two benchmark datasets, resulting in substantial performance loss.

We observe that compression causes greater performance degradation on M3-Bench-web than on M3-Bench-robot, further supporting our earlier hypothesis that the text nodes in M3-Bench-robot contain higher redundancy.

TMR Impact Analysis. As shown in Figure 4, when the TMR mechanism is not applied, the model's performance drops significantly on both benchmark datasets once the compression ratio exceeds 50%. In contrast, after introducing TMR, the model accuracy improves by **3.3%** over the original M3-Agent even without any compression. Compressing 30% of the text nodes further enhances the performance. More importantly, under a 70% compression setting, the model does not suffer from noticeable performance degradation and performs comparably to or even better than the uncompressed M3-Agent. These observations demonstrate that TMR mechanism can significantly reduce the accuracy loss caused by memory graph compression and effectively retrieve memories from more recent segments for reasoning.

Time Efficiency Analysis. In this section, we analyze the impact of the EMsampling module, EWpruning module, and TMR mechanism on the time efficiency of M3-Agent, with a focus on the memory retrieval time after converting the model query into embeddings. As shown in Figure 5, when only applying the EMsampling and EWpruning modules for compression, SS-Time decreases as the compression ratio increases, while Avg-Retrieval increases correspondingly (M3-Agent triggers up to five memory retrievals per question by default). We attribute

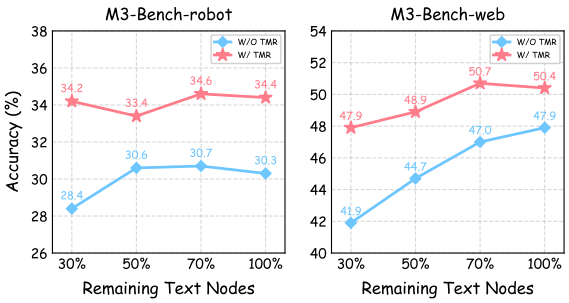


Figure 4: Impact of the TMR Mechanism on M3-Bench-robot and M3-Bench-web at different compression ratios.

this phenomenon to the reduced memory capacity after compression, which limits the amount of effective information the model can retrieve, lowers the confidence in the retrieval results, and requires more retrieval rounds to finalize the answer.

In contrast, with the introduction of the TMR mechanism, Avg-Retrieval significantly decreases, as TMR mechanism prioritizes more reliable and higher-confidence memory entries, enabling the model to obtain sufficient information with fewer retrieval iterations, thus effectively reducing the overall processing time.

When compressing **70%** of the memory, our method achieves an average **1.87**× speedup in memory retrieval time across all datasets.

Effect of Cluster Ratio *a*. As shown in Figure 6(a), the model performs best on both datasets when the cluster ratio $a = 0.05$. When a is smaller, the number of clusters decreases, and each cluster contains more samples, which makes it more likely that text nodes with large semantic differences are assigned to the same cluster, thereby affecting the preservation of semantic distinctions. In contrast, when a is larger, the number of clusters increases, and each cluster contains fewer samples, making it difficult to form stable and representative semantic structures, ultimately degrading the performance of the compressed model.

Effect of Balance Coefficient *b*. As shown in Figure 6(b), the model achieves the best performance on both datasets when the parameter balance coefficient $b = 0.1$. When b is further decreased or increased, the performance drops to varying degrees. This observation indicates that an excessively small balance coefficient prevents the model from effectively leveraging entity-importance information, whereas an excessively large balance coefficient causes the

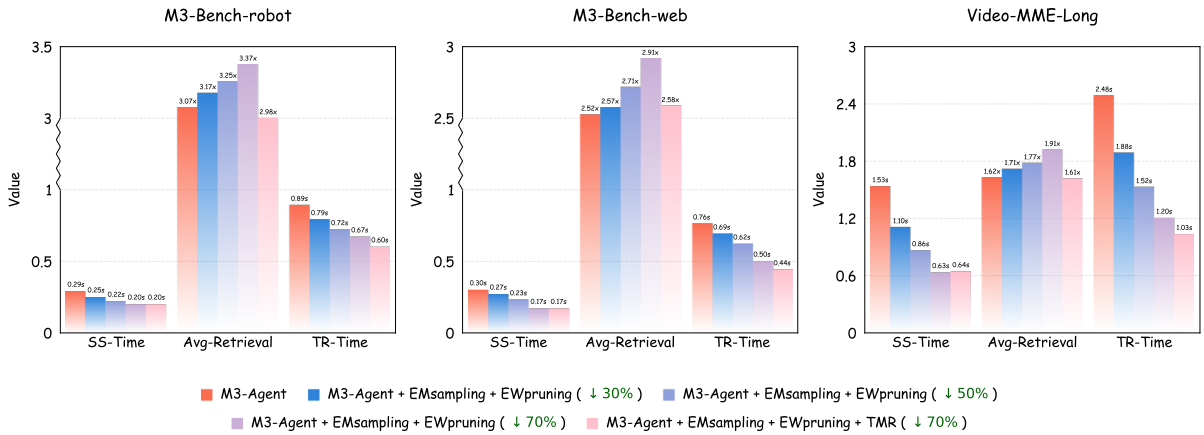


Figure 5: Time Efficiency Analysis on M3-Bench-Robot, M3-Bench-Web and Video-MME-Long. **SS-Time:** Time to compute and sort the similarity between the query embedding and each text node's content embedding; **Avg-Retrieval:** Average number of memory graph queries per question; **TR-Time:** Average total time spent querying the memory graph for each question after obtaining the model's query embedding.

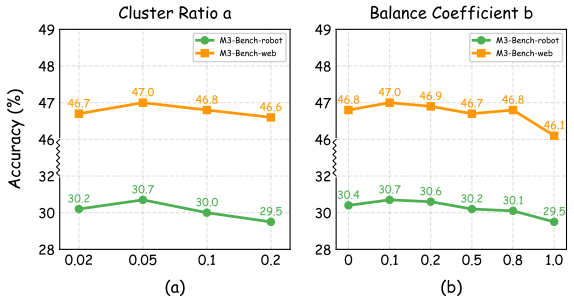


Figure 6: Impact of cluster ratio a and balance boeffi-cient b on M3-Bench-robot and M3-Bench-web.

model to overly rely on entity importance, thereby ignoring embedding similarity information and ultimately degrading the overall performance.

On Other Decay Methods We first compare linear decay with the exponential decay used in our paper (evaluated on the M3-Bench-Robot dataset). The results are shown in Table 3, where the parameter denotes the decay coefficient. From the table, exponential decay shows a generally monotonic decreasing trend, while linear decay exhibits a rise-then-fall pattern. This observation aligns with intuition: exponential decay decreases more rapidly and better matches the human memory forgetting curve, achieving stronger performance under smaller decay coefficients. We also note that linear decay could theoretically approximate exponential decay, but would require more delicate parameter tuning. Therefore, exponential decay is more practical and robust in real-world scenarios.

In addition, we experimented with a piecewise decay strategy: starting from the time segment corresponding to the query, the memory strength is halved every five temporal segments. This strategy achieves an accuracy of 33.8% on M3-Bench-Robot, which is still lower than exponential decay.

In addition, for more experiments and theoretical analysis, please refer to the appendix.

Decay Coefficient	Exponential	Linear
0.1	34.6	33.5
0.5	32.1	33.7
1.0	32.9	32.4
2.0	32.0	33.5

Table 3: Comparison between diffenent decay methods on M3-Bench-Robot.

6 Conclusion

We propose **StreamMeCo**, the first memory compression framework specifically designed for industrial-scale streaming video agent. It achieves efficient compression and accurate retrieval of memory graphs through the integration of our EMsampling and EWpruning modules, along with the TMR mechanism. Experimental results show that even with a **70%** reduction in memory graph size, StreamMeCo achieves a **1.87**× speedup in memory retrieval, while delivering an average accuracy improvement of **1.0%**.

Acknowledge

This project was supported by New Generation Artificial Intelligence-National Science and Technology Major Project.

Limitations

Due to the frequent need to call the Gemini-2.5-Pro and text-embedding-3-large APIs during memory graph generation, as well as the substantial time required to generate the memory graphs, and since M3-Agent currently only constructs and provides memory graphs for M3-Bench-robot and M3-Bench-web, we were limited by budget and time constraints and were only able to test the Video-MME-Long dataset additionally. We hope that future work can test more benchmark datasets to more comprehensively validate the performance of our method.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

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

Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. 2025. Streamingtom: Streaming token compression for efficient video understanding. *arXiv preprint arXiv:2510.18269*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025a. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025b. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Inderjit S Dhillon and Dharmendra S Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42(1):143–175.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. 2025. Streaming video question-answering with in-context video kv-cache retrieval. *arXiv preprint arXiv:2503.00540*.

Xingjian Diao, Tianzhen Yang, Chunhui Zhang, Weiyi Wu, Ming Cheng, and Jiang Gui. 2025a. Learning sparsity for effective and efficient music performance question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 136–146.

Xingjian Diao, Chunhui Zhang, Keyi Kong, Weiyi Wu, Chiyu Ma, Zhongyu Ouyang, Peijun Qing, Soroush Vosoughi, and Jiang Gui. 2025b. Soundmind: Rl-incentivized logic reasoning for audio-language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 528–540.

Xingjian Diao, Chunhui Zhang, Tingxuan Wu, Ming Cheng, Zhongyu Ouyang, Weiyi Wu, and Jiang Gui. 2024. Learning musical representations for music performance question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2803–2813.

Xingjian Diao, Chunhui Zhang, Weiyi Wu, Zhongyu Ouyang, Peijun Qing, Ming Cheng, Soroush Vosoughi, and Jiang Gui. 2025c. Temporal working memory: Query-guided segment refinement for enhanced multimodal understanding. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3393–3409.

Vaggelis Dorovatas, Soroush Seifi, Gunshi Gupta, and Rahaf Aljundi. 2025. Recurrent attention-based token selection for efficient streaming video-llms. *arXiv preprint arXiv:2510.17364*.

Yue Fan, Xiaojian Ma, Rongpeng Su, Jun Guo, Rujie Wu, Xi Chen, and Qing Li. 2025. Embodied videoagent: Persistent memory from egocentric videos and embodied sensors enables dynamic scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6342–6352.

Zhen Fang, Zhuoyang Liu, Jiaming Liu, Hao Chen, Yu Zeng, Shiting Huang, Zehui Chen, Lin Chen, Shanghang Zhang, and Feng Zhao. 2025. Dualvla: Building a generalizable embodied agent via partial decoupling of reasoning and action. *arXiv preprint arXiv:2511.22134*.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, and 1 others. 2025. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24108–24118.

Ruiyan Han, Zhen Fang, XinYu Sun, Yuchen Ma, Ziheng Wang, Yu Zeng, Zehui Chen, Lin Chen, Wenxuan Huang, Wei-Jie Xu, and 1 others. 2026. Unicorn: Towards self-improving unified multimodal models through self-generated supervision. *arXiv preprint arXiv:2601.03193*.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. 2024. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13504–13514.

Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. 2025a. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26683–26692.

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Kumara Kahatapitiya, Kanchana Ranasinghe, Jongwoo Park, and Michael S Ryoo. 2025. Language repository for long video understanding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5627–5646.

Minsoo Kim, Kyuhong Shim, Jungwook Choi, and Simyung Chang. 2025. Infinipot-v: Memory-constrained kv cache compression for streaming video understanding. *arXiv preprint arXiv:2506.15745*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Chenghao Li, Chaoning Zhang, Yi Lu, Jiaquan Zhang, Qigan Sun, Xudong Wang, Jiwei Wei, Guoqing Wang, Yang Yang, and Heng Tao Shen. 2025a. Syzygy of thoughts: Improving llm cot with the minimal free resolution. *arXiv preprint arXiv:2504.09566*.

Ruanjun Li, Yuedong Tan, Yuanming Shi, and Jiawei Shao. 2025b. Videoscan: Enabling efficient streaming video understanding via frame-level semantic carriers. *arXiv preprint arXiv:2503.09387*.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, and 1 others. 2025c. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. 2025. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. *arXiv preprint arXiv:2508.09736*.

Zhicong Lu, Haijun Xia, Seongkook Heo, and Daniel Wigdor. 2018. You watch, you give, and you engage: a study of live streaming practices in china. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13.

Xueqi Ma, Shuo Yang, Yanbei Jiang, Shu Liu, Zhenzhen Liu, Jiayang Ao, Xingjun Ma, Sarah Monazam Erfani, and James Bailey. 2026. Attention in space: Functional roles of vlm heads for spatial reasoning. *arXiv preprint arXiv:2603.20662*.

Mohammad Reza Ghasemi Madani, Soyeon Caren Han, Shuo Yang, and Jey Han Lau. 2026. Inclusion-of-thoughts: Mitigating preference instability via purifying the decision space. *Preprint*, arXiv:2604.04944.

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

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. 2025. Dispider: Enabling video llms with active real-time interaction via disentangled perception, decision, and reaction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24045–24055.

Aravinda S Rao, Marko Radanovic, Yuguang Liu, Songbo Hu, Yihai Fang, Kourosh Khoshelham, Marimuthu Palaniswami, and Tuan Ngo. 2022. Real-time monitoring of construction sites: Sensors, methods, and applications. *Automation in construction*, 136:104099.

Yuhao Shen, Tianyu Liu, Junyi Shen, Jinyang Wu, Quan Kong, Li Huan, and Cong Wang. 2026. Double: Breaking the acceleration limit via double retrieval speculative parallelism. *arXiv preprint arXiv:2601.05524*.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, and 1 others. 2024. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18221–18232.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. 2025. Adaptive keyframe sampling for long video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29118–29128.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer,

Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

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

Mengyue Wang, Shuo Chen, Kristian Kersting, Volker Tresp, and Yunpu Ma. 2025b. Metok: Multi-stage event-based token compression for efficient long video understanding. *arXiv preprint arXiv:2506.02850*.

Tongxi Wang. 2026. Fbs: Modeling native parallel reading inside a transformer. *arXiv preprint arXiv:2601.21708*.

Xudong Wang, Chaoning Zhang, Jiaquan Zhang, Chenghao Li, Qigan Sun, Sung-Ho Bae, Peng Wang, Ning Xie, Jie Zou, Yang Yang, and 1 others. 2026. Efficient and interpretable multi-agent llm routing via ant colony optimization. *arXiv preprint arXiv:2603.12933*.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. 2024. Memoryllm: Towards self-updatable large language models. *Preprint*, arXiv:2402.04624.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. 2025. Stop looking for important tokens in multimodal language models: Duplication matters more. *arXiv preprint arXiv:2502.11494*.

Jinyang Wu, Shuo Yang, Changpeng Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026. Spark: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. *arXiv preprint arXiv:2601.20209*.

Haomiao Xiong, Zongxin Yang, Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Jiawen Zhu, and Huchuan Lu. 2025. Streaming video understanding and multi-round interaction with memory-enhanced knowledge. *arXiv preprint arXiv:2501.13468*.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, and 1 others. 2025a. Qwen2. 5-omni technical report. *arXiv preprint arXiv:2503.20215*.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Kelly Peng, Yao Lu, and Song Han. 2025b. Streamingvlm: Real-time understanding for infinite video streams. *arXiv preprint arXiv:2510.09608*.

Haolin Yang, Feilong Tang, Lingxiao Zhao, Xiang An, Ming Hu, Huifa Li, Xinlin Zhuang, Yifan Lu, Xiaofeng Zhang, Abdalla Swikir, and 1 others. 2025a. Streamagent: Towards anticipatory agents for streaming video understanding. *arXiv preprint arXiv:2508.01875*.

Shuo Yang, Caren Han, Siwen Luo, and Eduard Hovy. 2025b. Magic-vqa: Multimodal and grounded inference with commonsense knowledge for visual question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16967–16986.

Shuo Yang, Soyeon Caren Han, Yihao Ding, Shuhe Wang, and Eduard Hoy. 2026a. Tooltree: Efficient llm agent tool planning via dual-feedback monte carlo tree search and bidirectional pruning. *arXiv preprint arXiv:2603.12740*.

Shuo Yang, Soyeon Caren Han, Xueqi Ma, Yan Li, Mohammad Reza Ghasemi Madani, and Eduard Hovy. 2026b. Evotool: Self-evolving tool-use policy optimization in llm agents via blame-aware mutation and diversity-aware selection. *arXiv preprint arXiv:2603.04900*.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. 2025c. Streammem: Query-agnostic kv cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*.

Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, and 1 others. 2025a. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10807–10816.

Yiqun Yao, Naitong Yu, Xiang Li, Xin Jiang, Xuezhi Fang, Wenjia Ma, Xuying Meng, Jing Li, Aixin Sun, and Yequan Wang. 2025b. Egomem: Lifelong memory agent for full-duplex omnimodal models. *arXiv preprint arXiv:2509.11914*.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025a. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

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

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, and 1 others. 2025. Streamforest: Efficient online video understanding with persistent event memory. *arXiv preprint arXiv:2509.24871*.

Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, and 1 others. 2026. Vision-deepresearch benchmark: Rethinking visual and textual search for multimodal large language models. *arXiv preprint arXiv:2602.02185*.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, Jifeng Dai, and Xiaojie Jin. 2024. Flash-vstream: Memory-based real-time understanding for long video streams. *arXiv preprint arXiv:2406.08085*.

Jiaquan Zhang, Qigan Sun, Chaoning Zhang, Xudong Wang, Zhenzhen Huang, Yitian Zhou, Pengcheng Zheng, Chi lok Andy Tai, Sung-Ho Bae, Zeyu Ma, Caiyan Qin, Jinyu Guo, Yang Yang, and Hengtao Shen. 2026a. Tda-rc: Task-driven alignment for knowledge-based reasoning chains in large language models. *Preprint*, arXiv:2604.04942.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Yibei Liu, Chenghao Li, Qigan Sun, Shuai Yuan, Fachrina Dewi Puspitasari, Dongshen Han, Guoqing Wang, and 1 others. 2026b. Text summarization via global structure awareness. *arXiv preprint arXiv:2602.09821*.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Xudong Wang, Zhenzhen Huang, Pengcheng Zheng, Shuai Yuan, Sheng Zheng, Qigan Sun, Jie Zou, and 1 others. 2026c. Learning global hypothesis space for enhancing synergistic reasoning chain. *arXiv preprint arXiv:2602.09794*.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

Jialong Zuo, Yongtai Deng, Lingdong Kong, Jingkang Yang, Rui Jin, Yiwei Zhang, Nong Sang, Liang Pan, Ziwei Liu, and Changxin Gao. 2025. Videolucy: Deep memory backtracking for long video understanding. *arXiv preprint arXiv:2510.12422*.

Appendix

A Other Experimental Details

A.1 Dataset

The detailed statistics of the three benchmark datasets are provided in Table 4, and the detailed information of the memory graph is shown in Figure 7. Next, we provide a detailed description of each dataset.

M3-Bench-robot. M3-Bench-robot contains 100 real-world videos recorded from a robot's first-person perspective, covering seven everyday environments, such as living rooms, kitchens, bedrooms, study rooms, offices, meeting rooms, and gyms. Each video depicts interactions between the robot and two to four different humans, where the robot must build and reason based on these interactions. The dataset includes five reasoning types: ME (multi-evidence reasoning), MH (multi-hop reasoning), CM (cross-modal reasoning), PU (person understanding), and GK (general knowledge extraction). M3-Bench-robot

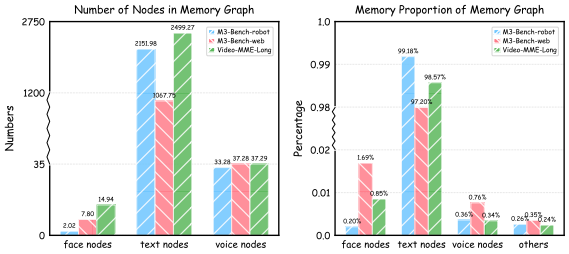


Figure 7: Numbers of different types of nodes and their memory proportion in the memory graphs of the three benchmark datasets.

aims to evaluate how robots use long-term memory to reason, particularly focusing on complex spatial relationships, object tracking, and dynamic human interactions.

M3-Bench-web. M3-Bench-web contains 920 videos sourced from YouTube, covering 46 different video types such as documentaries, street interactions, variety shows, travel, food, and sports events. The videos provide a wide range of content, designed to challenge the agent's ability to reason in complex and dynamic environments. Like M3-Bench-robot, M3-Bench-web also includes five reasoning types, assessing how the agent integrates cross-modal information, understands relationships between people, and extracts general knowledge. This dataset is ideal for evaluating the agent's memory reasoning capabilities in diverse tasks and complex scenarios, especially for cross-modal reasoning and memory retrieval in varied contexts.

Video-MME-Long. Video-MME-Long is the long-video subset of the Video-MME benchmark, consisting of 300 real-world long videos and 900 multiple-choice questions. The videos have an average duration of approximately 41 minutes and may extend up to 1 hour, covering 6 major visual domains and 30 fine-grained categories. The dataset provides multimodal inputs including video frames, audio, and subtitles, while the questions emphasize cross-temporal event association and complex spatiotemporal reasoning. This makes Video-MME-Long a highly challenging benchmark for evaluating multimodal large language models in long-term temporal understanding and multimodal information integration.

A.2 Baselines

A.2.1 Closed-source MLLMs

Gemini-1.5-Pro. Gemini 1.5 Pro is an advanced multimodal model capable of handling long-

Dataset	Avg. Duration(s)	#Videos	#QA Pairs	Avg. Text Nodes	#Source
M3-Bench-robot	2039.9	100	1276	2151.98	Real-world filming
M3-Bench-web	1630.7	920	3214	1067.75	YouTube
Video-MME-Long	2466.7	300	900	2499.27	YouTube

Table 4: The statistical information of M3-Bench-robot, M3-Bench-web and Video-MME-Long.

contexts of up to 10 million tokens. It is based on a sparse mixture-of-experts (MoE) Transformer architecture, efficiently processing various data types such as text, video, and audio, and performs exceptionally well across multiple benchmarks.

GPT-4o. GPT-4o is an advanced multimodal model capable of processing a variety of input types, including text, images, and videos. It excels in real-time response, efficiently handling inputs and generating multimodal outputs. The system is particularly optimized for visual understanding, enabling fast and accurate interactions in multimodal tasks. This design allows GPT-4o to perform robustly across diverse challenging multimodal benchmarks.

A.2.2 Open-source MLLMs

Qwen2.5-VL. Qwen2.5-VL is a powerful multimodal vision-language model designed for understanding and reasoning across various input types, including images, videos, and text. It excels in tasks such as precise object localization, document parsing, and long-video comprehension, with breakthroughs in video understanding and object localization. Qwen2.5-VL also supports efficient reasoning across diverse domains, handling complex input data and generating accurate multimodal outputs.

Qwen2.5-Omni. Qwen2.5-Omni is a unified multimodal model capable of processing various inputs such as text, images, audio, and video, while simultaneously generating text and natural speech responses in a streaming manner. It utilizes an innovative Thinker–Talker architecture, enabling real-time speech responses, and excels in tasks like audio understanding and video reasoning, making it suitable for real-time multimodal interaction scenarios.

A.2.3 Streaming Video Models

MovieChat. MovieChat is a video understanding system that integrates video models and LLM. It uses a sliding window to extract frame-level features and stores them in hybrid memory, with the LLM performing QA based on this memory. Supporting long videos over 10K frames, the system effectively addresses computational complexity and memory overhead, enhancing overall long video comprehension.

MA-LMM. MA-LMM is a memory-augmented large multimodal model designed for long-term video understanding. It processes video frames sequentially and stores information in a long-term memory bank, avoiding the context length and GPU memory limitations faced by traditional models when handling long videos. The model efficiently captures temporal information in videos and achieves excellent performance in tasks such as video question answering and video captioning.

VideoChat-Online. VideoChat-Online is an efficient video LLM designed for streaming video understanding, utilizing a Pyramid Memory Bank (PMB) architecture to balance spatial and temporal details. The model is optimized through an offline-to-online learning paradigm, combined with an interleaved dialogue format for video data, enabling real-time video stream processing and outstanding performance across multiple benchmarks. It outperforms many existing models in online video tasks and demonstrates excellent cross-platform capabilities.

Flash-VStream. Flash-VStream is a real-time video stream understanding model that employs the STAR (Spatial-Temporal-Abstract-Retrieved) memory mechanism, enabling efficient processing of extremely long video streams and real-time user query responses. By compressing visual information and dynamically updating memory, the model significantly reduces inference latency and GPU memory consumption, supporting online video question answering. Flash-VStream outperforms existing methods in multiple benchmarks, especially in streaming video scenarios.

TimeChat-Online. TimeChat-Online is an advanced online VideoLLM designed for real-time video interaction. Its core innovation lies in the

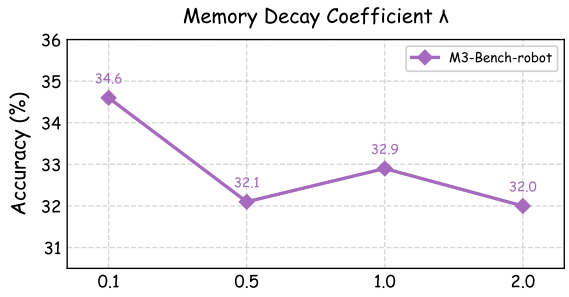


Figure 8: Impact of memory decay coefficient λ on M3-Bench-robot.

Differential Token Drop (DTD) module, which reduces visual redundancy in streaming videos by selectively preserving significant temporal changes while discarding static content between frames. This approach helps achieve an 82.8% reduction in video tokens, maintaining high performance in real-time video question answering (QA). Additionally, it integrates proactive responding capabilities, triggering answers based on future visual cues from video scene transitions.

StreamForest. StreamForest is an online video understanding model specifically designed for streaming video scenarios. Its core innovation lies in the Persistent Event Memory Forest, which dynamically organizes video frames into event-level tree structures, ensuring effective storage and processing of long-term visual information. The model also introduces a Fine-grained Spatiotemporal Window, enhancing real-time scene perception. It performs exceptionally well across multiple streaming video understanding benchmarks, particularly in real-time video question answering and autonomous driving applications.

M3-Agent. M3-Agent is the first industrial-grade streaming video agent, integrated with a long-term Agent Memory mechanism, capable of processing video and audio inputs in real-time. It generates episodic and semantic memories to gradually build world knowledge and completes tasks through multi-turn reasoning.

A.3 Baselines setup

A.4 Closed-source MLLMs

M3-Agent consists of two models, memorization and control, and is trained using reinforcement learning. Following the approach in the M3-Agent paper, we use the closed-source MLLMs Gemini-1.5-Pro and GPT-4o to replace the memorization model, generating descriptions for video clips and storing them as long-term memory. Subsequently, through retrieval-augmented generation (RAG) (Lewis et al., 2020), we use GPT-4o to perform memory retrieval as a replacement for the control model.

Following the approach in the M3-Agent paper, for Gemini-Agent, we use Gemini-1.5-Pro to replace the memorization and control models in M3-Agent. For the generated memory graph, we use M3-Agent's search function to replace RAG. Similarly, for Gemini-GPT4o-Hybrid, we use Gemini-1.5-Pro to generate memory as the memory model, use GPT-4o for control, and still use M3-Agent's search function to replace RAG.

A.4.1 Open-source MLLMs

Following the approach in the M3-Agent paper, we use the open-source MLLMs Qwen2.5-VL and Qwen2.5-Omni to replace the closed-source MLLMs Gemini-1.5-Pro and GPT-4o in Section A.4, to ensure a fair comparison, with the rest remaining consistent as before.

A.5 Streaming Video Models

For the streaming video models, we use the pre-trained models and default parameter settings from the official papers.

B Additional Ablation Study

B.1 Effect of Memory Decay Coefficient λ.

As shown in Figure 8, the model achieves its best performance when the memory decay coefficient is set to memory decay coefficient λ = ***0.1***. As λ increases further, the performance drops to varying degrees. This indicates that an excessively large decay coefficient leads to an overly strong temporal decay effect, causing the model to diminish the importance of historical memory too early. Consequently, the model fails to effectively leverage long-term contextual information, ultimately degrading its reasoning performance.

B.2 TMR Impact on Video-MME-Long

As shown in Figure 9, consistent with the trends observed in M3-Bench-robot and M3-Bench-web, the model's performance on the Video-MME-Long dataset significantly decreases when the compression ratio exceeds 50%. After introducing the TMR mechanism, the performance degradation is effectively mitigated, and even the performance at a 50% compression ratio surpasses

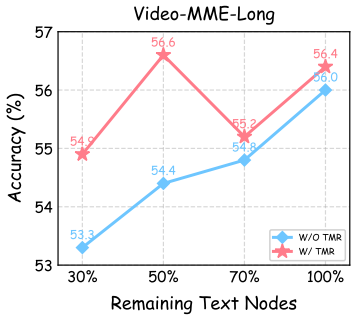


Figure 9: Impact of the TMR Mechanism on Video-MME-Long at different compression ratios.

Components			M3-Bench-Robot	M3-Bench-Web
A	B	C	Accuracy	Accuracy
–	–	–	30.3 (100.0%)	47.9 (100.0%)
			26.6 (87.8%)	41.8 (87.3%)
✓			27.0 (89.1%)	42.6 (88.9%)
	✓	✓	28.7 (94.7%)	43.9 (91.6%)
✓	✓		28.4 (93.7%)	42.3 (88.3%)
✓		✓	28.9 (95.4%)	44.2 (92.3%)
✓	✓	✓	**30.6 (101.0%)**	**44.7 (93.3%)**

Table 5: Ablation study of the compression components. The first row reports the accuracy of the M3-Agent without compression. **A**: EMsampling; **B**: entity importance; **C**: embedding similarity.

that at 30%. These results are in strong agreement with the previous experiments, indicating that the TMR mechanism can significantly reduce the accuracy loss caused by memory graph compression and retrieve more critical memories for reasoning.

B.3 Experiments with Other Compression Ratios

Previously, to highlight the contribution of each module, we conducted experiments under a 30% compression ratio, and the corresponding results are reported in Table 2 of the paper. Here, we further provide ablation results under a 50% compression ratio, as shown below in Table 5 (A: EMsampling; B: entity importance; C: embedding similarity). As observed, compared with random compression, our method still achieves consistent improvements even without the TMR mechanism (i.e., without TMR).

C Time Complexity Analysis

Our compression framework consists of two main modules: EMsampling and EWpruning. Suppose the memory graph contains N nodes in total. The EMsampling module includes KMeans clustering and Minmax sampling. The clustering stage has a time complexity of $O(N \log N)$, while Minmax sampling requires computing pairwise distances between nodes, resulting in a time complexity of $O(N^2)$. Therefore, the overall complexity of EMsampling is $O(N^2)$. For the EWpruning module, the main computational cost comes from constructing the similarity matrix and the weighted edge matrix, both of which involve pairwise node interactions, leading to a time complexity of $O(N^2)$. In summary, the overall time complexity of our method is $O(N^2)$. However, in practical scenarios, for streaming video memory graphs covering 30–60 minutes, each compression run takes less than 2 seconds, demonstrating high efficiency in real-world settings.

D Theoretical Analysis

Within a semantic cluster C, let the original node embeddings be $X = \{x_i\}_{i=1}^{N}$. EMsampling adopts a minmax rule to iteratively select a subset $S \subseteq X$, where the distance is defined as $d(x, S) = \min_{y \in S} \|x - y\|$. At each step, the algorithm selects the node that is hardest to cover (i.e., the one maximizing $d(x, S)$) and adds it to S. This greedy process is equivalent to progressively shrinking the covering radius $\rho = \max_{x \in X} d(x, S) = \max_{x \in X} \min_{y \in S} \|x - y\|$. When ρ is small, every original node x has a nearby representative in S within distance ρ. As a result, the compressed nodes uniformly cover the original cluster in embedding space, avoiding semantic holes and theoretically leading to minimal accuracy degradation.

For the EWpruning module, we analyze the fused importance score r. Let the total scores before and after compression be X_1 and X_2. According to the scoring function $r_j = bW'_j + (1 - b)S'_j$, nodes with smaller r_j are preferentially removed. Let the number of nodes before and after compression be N and M. The score loss satisfies $\Delta X = X_1 - X_2 = \sum_{i=1}^{N} r_i - \sum_{j=1}^{M} r_j \approx 0$. Therefore, the information loss introduced by compression is theoretically minimal.

E Experiments on Other Graph-Based Memory Frameworks

Our method can be readily adapted to other graph-based Agent Memory frameworks. Specifically, we adapt it to the Mem0 (graph) (Chhikara et al., 2025b) framework, whose memory paradigm includes entity types (e.g., persons, events), embed-

Model / Dataset	Office (95)	Meeting Room (68)	Total (163)
Mem0 (graph)	42.1	45.6	43.6
+Random	33.7	36.8	35.0
+Clustering	35.8	39.7	38.0
+DART	34.7	42.6	38.0
+TimeChat-Memory	37.9	39.7	38.7
+MemoryLLM	34.7	38.2	36.2
+StreamMeCo (w/o TMR)	40.0	44.1	41.7

Table 6: Mem0 (graph): Performance comparison of different compression methods on the Office and Meeting Room subsets of M3-Bench-robot (compression ratio = 30%).

dings, textual memory content, and timestamps. In this setting, nodes connected to entity types such as "person" are treated as connected nodes, and we apply the EWpruning module (since this framework does not provide edge weights, we set them to 1). The remaining nodes are treated as isolated textual nodes.

Currently, M3-Agent is the only graph-based Agent Memory framework capable of handling streaming video inputs, while most other frameworks focus on textual inputs. Therefore, we convert the memory graph generated by M3-Agent into textual memories and feed them into Mem0 (graph) for evaluation. We conduct experiments on the office and meeting-room subsets of the M3-Bench-robot dataset. This subset originates from real-world streaming videos and contains 14 videos and 163 questions, making it closer to real-time monitoring scenarios.

The results (compression ratio = 30%) are shown in Table 6. Our method remains effective in other graph-based Agent Memory frameworks and achieves the best performance. As long as a graph-based Agent Memory framework defines entities such as persons, our method can be directly applied. Moreover, when adapted to other graph-based Agent Memory frameworks, our approach can be seamlessly transferred to different domains, such as long-horizon dialogue scenarios.

F Future Works

Future work can further extend the ideas proposed in this study. First, this work adopts a unified compression ratio for isolated text nodes and connected text nodes; however, future research could explore differentiated and adaptive compression strategies for different node types, so as to better exploit their structural characteristics and information distributions. Second, future studies may investigate the differential compression of episodic nodes and semantic nodes, where episodic nodes primarily focus on specific events occurring in video segments, while semantic nodes emphasize general and relatively stable knowledge distilled from these segments. Effectively preserving highly discriminative event-level descriptions and critical stable semantic memories during compression is an important direction for further enhancing the expressive power and reasoning support of the memory graph. In addition, we observe a noticeable redundancy among text nodes in the M3-Bench-robot dataset. Future work may systematically study the identification and resolution of redundant or even conflicting memories in the memory graph, as well as potential memory poisoning issues. By selectively removing or correcting unreliable memories, the robustness and reliability of the model in complex streaming video scenarios can be further improved. Finally, reducing the time overhead required for memory graph construction in the early stage is also an important research direction. Future work may explore more efficient memory construction mechanisms to lower the cost of generating memory graphs.