# RMem: Restricted Memory Banks Improve Video Object Segmentation

## Supplementary Material

Our supplementary materials cover additional analysis, implementation details, and discussion as below:

(A) **Demo Video.** We provide a demo video at https://youtu.be/mFjGSPXmXdA showing multiple challenging VOS examples (Sec. A).

(B) **Implementation details.** We explain the detailed model architectures and the procedures for training and inference (Sec. B).

(C) **Additional ablation studies.** This section provides more analysis and experimental results (Sec. C).

(D) **Addition discussion on limitations and future work.** We offer a more detailed discussion of the limitations and potential future directions (Sec. D).

## A. Demo Video

In https://youtu.be/mFjGSPXmXdA, we provide four qualitative comparison examples between the baseline models (AOT [17] and DeAOT [15]) and our RMem, with the object state changes from both VOST [12] and the Long Videos dataset [7]. Notably, these examples illustrate four challenging scenarios: (1) **Object ambiguity**: objects have similar appearances; (2) **Slicing**: an object is cut into multiple slices; (3) **Appearance changes**: an object has changed its shape and appearances, leading to incorrect VOS masks. (4) **Sudden shape changes**: the viewpoint changes quickly and causes variation in shapes of the target object. The four examples demonstrate that RMem effectively improves the spatio-temporal reasoning of VOS.

## B. Implementation Details

We describe the outline of implementation of AOT [17] and DeAOT [15] baselines in Sec. 5.2 (main paper). This section provides in-depth details of the implementation and the configuration of RMem.

### B.1. Model Architecture

AOT and DeAOT share the common architecture of the memory-based VOS framework. As conceptualized in Eqn. 1 (main paper), we disassemble the VOS framework into the modules of an encoder $\mathbf{E}(\cdot)$ encoding images into feature maps, a decoder $\mathbf{D}(\cdot)$ extracting information from the memory bank, and a segmentation head translating the output from decoder into masks. Please note that we have additionally decoupled the segmentation head from the decoder for clarity, compared with Eqn. 1 (main paper).

**Encoder.** Identical to VOST [12], we adopt ResNet-50 [4] as the encoder, which achieves competitive performance while efficient enough to operate on Long Videos. The multiple stages in the ResNet encoder produce 3 levels of feature maps $\{F^4, F^8, F^{16}\}$ with 1/4, 1/8, and 1/16 the resolution of the original input image, respectively. Following the practice of AOT and DeAOT, the deepest feature map $F^{16}$ is the input to the decoder for memory reading, and $\{F^4, F^8\}$ are provided to the segmentation head as input for predicting high-quality masks.

**Decoder.** AOT and DeAOT utilize a specially-designed transformer [13] to conduct associative memory reading, named "Long Short-term Transformer" (LSTT). LSTT comprises three consecutive transformer layers to enhance features in the current frame with the memory bank. Adopting the same notations as Eqn. 1 (main paper), we conceptually illustrate this process as Eqn. A:

$$\begin{aligned} F_t^{(l+1)} = \mathrm{Attn}(\mathrm{Q} &= F_t^{(l)}, \\ \mathrm{K} &= \mathbf{M}^{(l)}[F_{0:t-1}], \\ \mathrm{V} &= \mathbf{M}^{(l)}[F_{0:t-1}]), \end{aligned} \tag{A}$$

where the superscript $(l)$ denotes the layer index of LSTT, ranging from 0 to 2. After the above process, We keep the implementation details identical to the original AOT and DeAOT. Please refer to them for more detailed configuration. Finally, the output feature $F_t^{(3)}$ replaces the feature map $F^{16}$ from the encoder not enhanced with spatio-temporal information.

**Segmentation Head.** To maintain high-resolution segmentation masks, the segmentation process involves a feature pyramid network (FPN) [8]. It accepts $F_t^{(4)}$ as the input feature, uses $\{F^8, F^{16}\}$ as shortcut inputs, and up-samples them via the combination of a convolutional layer and a bilinear up-sampling layer.

**Temporal Positional Embedding.** We introduce temporal positional embedding (TPE) in Sec. 4.3 (main paper) to enhance the spatio-temporal reasoning ability of models. In practice, we initialize end-to-end learnable embeddings with the same number to the memory length during the training time (*e.g.*, 4 in VOST) and the same dimension to the feature $F_t$, marking the PE of each place in the memory bank. For simplicity, the three LSTT layers in Eqn. A share the same set of TPE.

### B.2. Training

**Loss Functions.** Our training procedure utilizes the same loss functions as AOT and DeAOT: the combination of bootstrapped cross-entropy loss and soft Jaccard loss [10].

Both loss terms are averaged 1:1 as the final loss value.

**VOST.** The training on VOST [12] follows the original practice of VOST's authors, where the models are fine-tuned on VOST with pretrained weights from DAVIS2017 [11] and Youtube2019 [14]. As VOST highlights spatio-temporal modeling, we follow the authors' implementation of AOT by using a long sequence length of 15 frames during training and this accordingly enables 4 frames in the memory bank. It leverages exponential moving averages (EMA) for parameter updates to stabilize the training process. The whole training process uses AdamW [6, 9] optimizer, and lasts 20,000 steps with a batch size of 8, on 4×A40 GPUs. The initial learning rate is $2 \times 10^{-4}$ and it gradually decays to $2 \times 10^{-5}$ according to a polynomial pattern [16]. To avoid overfitting, we set the learning rate of the encoder as 0.1 of the other components. The weight decay is 0.07, which is also identical to AOT and DeAOT.

**Long Videos Dataset.** Following the standard practice [3, 7], we first train the AOT and DeAOT models on the DAVIS2017 [11] and YoutubeVOS2019 dataset [14], then conduct inference on the Long Videos dataset [7]. However, to support the training of positional embedding, we extend the length of training samples from the original 5 frames to 9 frames, to support 4 frames in the memory banks during the training time. Please note that we also re-train the baselines under the same setup to ensure a fair comparison. The training procedure leverages the similar optimization setting as described above for the VOST dataset, including the AdamW [6, 9] optimizer, weight decay of 0.07, polynomial learning rate decay [16] from $2 \times 10^{-4}$ to $2 \times 10^{-5}$, 0.1 scaling of the encoder learning rate, and EMA parameter updates. The only difference from VOST is training 100,000 steps with a batch size of 16, following the implementation of the original AOT and DeAOT on DAVIS2017 and YoutubeVOS2019 datasets.

## B.3. Inference.

**VOST.** Instead of appending features into memory at *a fixed frequency of 5 frames*, the authors of VOST developed a different strategy than on DAVIS2017 and YoutubeVOS2019 to address the CUDA memory issue caused by higher resolution and longer video duration: the memory bank is bounded by 30 frames and the frequency of updating memory banks is accordingly $L/30$, where $L$ is the length of the video. For our RMem, we follow the frequency of memory updates set by VOST, but bounds the size of memory banks to 9 frames, which is significantly smaller than the original cap of 30 frames. Therefore, our RMem needs to update the memory banks by removing the obsolete frames, and we describe the details of memory update in Sec. B.4 below.

**Long Videos Dataset.** When comparing to the other approaches on the Long Videos dataset (Table 2, main paper), we primarily rely on the VOS performance evaluated by XMem [3]. However, we re-implement the baselines of AOT and DeAOT for a fair comparison with RMem, since XMem has not released the code for evaluating both methods. Notably, *our re-implementation achieves better performance* compared to XMem's reported numbers. In practice, we determine the frequency of updating memory banks by $L/30$ to avoid CUDA memory issues, which is similar to the inference procedure on VOST. Our RMem shares the same inference setting as baseline, only restricting the memory bank size to 8 frames. Then, the memory update strategy is identical to VOST, as described in Sec. B.4.

## B.4. Memory Update

As is described in Sec. 4.2 (main paper), our RMem balances the relevance and freshness of frames in the memory bank using our algorithm inspired by UCB [2].

**Relevance.** As mentioned in Sec. 4.2, we use the attention scores from the transformers in the decoder (Eqn. 5, main paper) to reflect the relevance of a memory frame $R_k$. Since the LSTT decoder in AOT and DeAOT has three transformer layers, we intuitively select the attention scores from the 0-th transformer because it is closest to the original image embeddings $F_t$ and memory features $\mathbf{M}^t$ (ablation in Sec. C.3). To stabilize the relevance term and avoid fluctuations, we further apply the moving average technique to the relevance term. Suppose $R_k^{'}$ denotes the relevance values of a memory frame $k$ derived from the latest timestamp, the consequent relevance term $R_k$ is updated via:

$$R_k \leftarrow (1-\lambda)R_k^{'} + \lambda R_k, \tag{B}$$

where we set $\lambda = 0.8$ for both VOST and the Long Videos dataset. As we have noticed, using moving average for stabilization is a common technique for VOS on long videos, such as in AFB-URR [7].

**Freshness.** To balance the numerical scales of the relevance and freshness terms, we slightly modify Eqn. 4 (main paper) as below,

$$O_j = R_j + \alpha\sqrt{\frac{\log T}{t_j + B}}, \tag{C}$$

where $B$ smooths the numerical ranges of the freshness term, and $\alpha$ controls the individual contribution of relevance and freshness. In practice, we set $B = 8$ and $\alpha = 1.5$ for both VOST and the Long Videos dataset. Detailed ablation studies on the values of $\alpha$ are illustrated in Sec. C.2.

## C. Supplemental Ablation Studies

### C.1. Memory Update on the Long Videos Dataset

We analyze the memory update strategies on the Long Videos dataset [7] using our AOT baseline in Table A, in addition to the analysis on VOST [12] (Table 4, main paper). **(1)** Notably, we observe consistent improvement from our UCB-inspired memory update strategy combining both relevance and freshness of frames in the memory. **(2)** Similar to the results on VOST, our baseline of removing the 1-st frame in the memory has competitive performance but is inferior to our final UCB-inspired strategy. **(3)** The analysis in Table A also reveals several intriguing differences between the Long Videos dataset and VOST. Specifically, VOST highly relies on the relevance of frames and the reliable information from the 0-th frames because of its complexity in scenarios, while the Long Videos dataset highlights the utility of freshness of frames as a consequence of extremely long video duration.

| Method | Variants | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|---|
| Remove | 0-th | 88.1 | 86.3 | 89.9 |
| | 1-st | 88.3 | 86.6 | 90.1 |
| | Middle | 86.6 | 85.5 | 87.9 |
| | Latest | 85.4 | 84.1 | 86.7 |
| | Random | 87.7 | 86.6 | 88.9 |
| UCB | Relev | 86.9 | 85.4 | 88.3 |
| | Relev + Fresh | **89.5** | **87.8** | **91.2** |

Table A. Ablation study of different memory updating strategies on the Long Videos dataset, in addition to VOST (Table 4, main paper). We analyze deleting a frame in the memory based on heuristics ("Remove") or guided by the relevance and freshness of the UCB algorithm ("UCB"). Our final memory updating strategy using both relevance and freshness achieves the best performance.

### C.2. Balancing Relevance and Freshness

As mentioned in Sec. 4.2 (main paper) and Sec. B.4, we balance relevance and freshness when updating the memory banks via Eqn. C. Fig. A analyzes the performance under different $\alpha$ values on both VOST and the Long Videos dataset. Specifically, a larger $\alpha$ denotes relying more on the freshness term. A proper $\alpha$ is essential for the UCB-inspired algorithm to improve memory update for both VOST and the Long Videos dataset, and we empirically select $\alpha = 1.5$ because it generalizes better to both of the datasets. Interestingly, Fig. A also reveals the difference between VOST and the Long Videos dataset: VOST has more complex scenarios and highlights the utility of relevance, while the long video dataset relies more on freshness due to its extremely long video duration. Nonetheless, our final $\alpha = 1.5$ achieves proper balance for both domains.

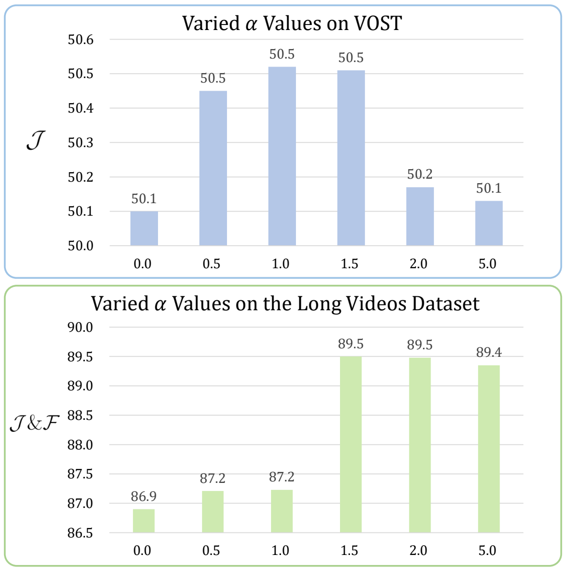

Figure A. Analysis on relevance and freshness for memory update, on VOST and the Long Videos dataset. The performance varies with different $\alpha$ values (from Eqn. C), and it illustrates the importance of the trade-off between relevance and freshness.

### C.3. Relevance Calculation

Our relevance term for memory update uses attention scores to reflect the importance of a frame, similar to previous works [3, 7]. However, LSTT has three transformer layers and enables two intuitive strategies of relevance calculation: (1) directly using the 0-th layer; and (2) computing the average attention scores of all the transformer layers. Table B compares these two strategies on VOST and the Long Videos dataset. We observe that using the 0-th layer for relevance calculation has an advantage in most of the scenarios. We conjecture that the 0-th transformer has the largest fidelity to the features of images and memory banks. Therefore, our RMem empirically selects the 0-th transformer for relevance, as described in Sec. B.4.

| Methods | VOST | | Long Video | | |
|---|---|---|---|---|---|
| | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
| AOT + RMem (0-th) | **39.8** | **50.5** | **90.3** | **88.5** | **92.1** |
| AOT + RMem (Mean) | 39.6 | 50.3 | 89.8 | 88.2 | 91.5 |
| DeAOT + RMem (0-th) | 40.4 | 51.8 | **91.5** | **89.8** | **93.3** |
| DeAOT + RMem (Mean) | **40.6** | **52.0** | 90.3 | 88.7 | 92.0 |

Table B. Analysis on the relevance calculation. "0-th" and "Mean" denote using the attention scores from the 0-th transformer layer or the average attention scores from all of the three layers.

### C.4. Analysis on Training-Inference alignment

As discussed in Sec. 4.3 (main paper), the purpose of temporal positional embedding is to align the gap between training and inference, as VOS models are trained on short videos but inferencing on unlimited videos. However, it is also valuable to explore whether it is another approach to address this training-inference gap. We compared our Restricted Memory (RM) with 2 approaches: **(1) Longer Memory (LM):** train the model with longer video clips so that the model can fit better on a larger memory bank. **(2) More Steps (MS):** train the model with more steps. As is shown in Table C, LM certainly is effective in mitigating the training-inference gap, but it is still worse than our RM. MS exhibits overfitting with too many training steps, thus not capable of addressing this issue. However, MS can still gain improvement through our RM, proving our method's effectiveness from another perspective.

| Model | Train_Mem_Len | Step | URM | | RM | |
|---|---|---|---|---|---|---|
| | | | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
| AOT | 4 | 20k | 37.0 | 49.2 | 38.6 | 50.2 |
| AOT-LM | 6 | 20k | 38.2 | 49.9 | 39.8 | 50.1 |
| AOT-MS | 4 | 40k | 36.6 | 48.6 | 37.8 | 48.0 |

Table C. Analysis of 2 approaches to address training-inference gap. "URM" for unrstricted memory and "RM" for restricted memory. Our "RM" is still the best way to align training and inference.

## C.5. Analysis on LVOS

Since the Long Videos dataset only features 3 testing videos, which is not able to fully demonstrate the effectiveness of our method, we further report our model's performance on LVOS dataset [5], which contains 50 long videos in the validation set.

| Methods | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|
| AOT | 63.6 | 57.6 | 69.5 |
| AOT + TPE | 64.5 | 58.9 | 70.0 |
| AOT + RMem | **66.1** | **60.5** | **71.7** |

Table D. Results on the validation set of LVOS dataset.

As is shown in Table D, our RMem still holds the highest performance compared to the AOT baseline. Besides, our TPE (temporal positional embedding) exhibits considerable improvements, which proves that TPE is effective in aligning the training-inference gap, given that the average duration in LVOS is much longer than other video datasets.

## C.6. Analysis on YoutubeVOS2019

Our study concentrates on improving the VOS accuracy for long and/or complex VOS scenarios. Meanwhile, we also supplement with analysis on shorter, simpler benchmarks. As indicated in Table 6 of the main paper, our RMem demonstrates comparable performance to baselines without RMem on DAVIS2017 [11], with a notable increase in efficiency. This result underlines the adaptability of our approach across different regimes.

Further analysis is conducted in the section using the YoutubeVOS2019 [14] benchmark, with shorter video duration and easier scenarios. In Table E, we evaluate two settings: (1) the influence of only restricting the memory bank sizes; and (2) the effect of the full RMem with temporal positional embedding. Table E (rows 1 and 2) shows that: by limiting the memory banks with the *original checkpoint* provided by DeAOT's authors, we maintain the same VOS quality. This finding suggests that *constraining the memory banks is a regime-independent strategy*.

A key aspect of our RMem is temporal positional embedding (TPE), which necessitates end-to-end model training on extended sequences. As in Sec. B.2, we *increase the training sequence length from 5 frames to 9 frames* without tuning the hyper-parameters, ensuring a 4-frame memory bank during the training stage. However, this introduces optimization challenges, as reflected in the decreased DeAOT performance with longer training clips (Table E, rows 1 and 3). Under such a setup and fair comparison, our full RMem has maintained comparable VOS quality compared with the baseline (rows 3 and 4). In conclusion, our RMem is also applicable for YoutubeVOS2019, although tuning the optimal hyper-parameters for training with longer sequence lengths is future work.

| Index | Method | $\mathcal{G}$ | $\mathcal{J}_s$ | $\mathcal{J}_u$ | $\mathcal{F}_s$ | $\mathcal{F}_u$ |
|---|---|---|---|---|---|---|
| 1 | DeAOT | 85.9 | 84.6 | 89.4 | 80.8 | 88.9 |
| 2 | DeAOT + RMem | 85.9 | 84.6 | 89.4 | 80.8 | 88.9 |
| 3 | DeAOT$^{\Psi}$ | 85.6 | 84.8 | 80.0 | 89.7 | 88.0 |
| 4 | DeAOT$^{\Psi}$ + RMem | 85.5 | 84.6 | 79.8 | 89.4 | 88.2 |

Table E. Analysis on YoutubeVOS2019 shows that, although not the primary focus of this paper, our RMem is also applicable for YoutubeVOS2019 with comparable performance with baselines. We first apply restricted memory banks to the original DeAOT checkpoint (rows 1 and 2). To enable temporal positional embedding (TPE), we train DeAOT under a longer sequence length and denote such models with "$\Psi$" (rows 3 and 4). The subscripts "s" and "u" denote the "seen" and "unseen" subsets of YoutubeVOS2019, respectively.

## D. Additional Discussion on Limitations and Future Work

We briefly outlined the limitations of our study in Sec. 6 due to space limits. This section elaborates on more details.

As mentioned in Sec. 6, we prioritize the analysis of memory banks, and RMem is designed as a straightforward instantiation to demonstrate our insight. For this purpose, our study primarily engages with state-of-the-art methods like AOT [17] and DeAOT [15]. This choice is grounded, especially when common VOS studies are built upon a single or few preceding approaches due to the complexity of the framework, such as XMem [3], HODOR [1], and DeAOT [15]. One potential limitation could be that our

RMem might implicitly depend on the transformer mechanisms and the affinity calculation in self-attention, which are adopted in AOT and DeAOT. These mechanisms natively support the temporal positional embedding and align with our key motivation of focusing the attention scores on relevant frames (Sec. 3 and Fig. 2, main paper). While future endeavors could explore adapting RMem for various VOS methods beyond the ones using transformers, near-future VOS methods will likely continue to employ a transformer-based framework, making our current RMem design compatible with them.

Another aspect mentioned in Sec. 6 is the potential for enhancing RMem with more advanced techniques. While the current simplicity of our approach effectively demonstrates our core insights into managing memory bank capacities, we acknowledge that it can benefit from a more sophisticated design. As especially pointed out in Sec. 6, XMem [3] exhibits an intricate design for efficiently expanding memory banks. Though more complex than our current method of simply bounding memory bank sizes, such advancements could offer greater flexibility and potentially improve VOS.

Lastly, as discussed in Sec. 6, another option for enhancement lies in improving the decoding capabilities of the VOS framework. Our study maintains the original design of existing methods for a fair comparison, yet future research could explore scaling or modifying VOS architectures to further mitigate the challenges posed by expanding memory banks.

## References

[1] Ali Athar, Jonathon Luiten, Alexander Hermans, Deva Ramanan, and Bastian Leibe. HODOR: High-level object descriptors for object re-segmentation in video learned from static images. In *CVPR*, 2022. 4

[2] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 2002. 2

[3] Ho Kei Cheng and Alexander G Schwing. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 2, 3, 4, 5

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[5] Lingyi Hong, Wenchao Chen, Zhongying Liu, Wei Zhang, Pinxue Guo, Zhaoyu Chen, and Wenqiang Zhang. LVOS: A benchmark for long-term video object segmentation. In *ICCV*, 2023. 4

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014. 2

[7] Yongqing Liang, Xin Li, Navid Jafari, and Jim Chen. Video object segmentation with adaptive feature bank and uncertain-region refinement. In *NeurIPS*, 2020. 1, 2, 3

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[10] Sebastian Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *CVPR*, 2014. 1

[11] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 2, 4

[12] Pavel Tokmakov, Jie Li, and Adrien Gaidon. Breaking the "object" in video object segmentation. In *CVPR*, 2023. 1, 2, 3

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[14] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-VOS: Sequence-to-sequence video object segmentation. In *ECCV*, 2018. 2, 4

[15] Zongxin Yang and Yi Yang. Decoupling features in hierarchical propagation for video object segmentation. In *NeurIPS*, 2022. 1, 4

[16] Zongxin Yang, Yunchao Wei, and Yi Yang. Collaborative video object segmentation by foreground-background integration. In *ECCV*, 2020. 2

[17] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 1, 4