SAM3RL: MEMORY CONTROL VIA REINFORCEMENT LEARNING FOR VISUAL OBJECT TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Segment Anything Model 3 (SAM 3) is a foundation model for image and video segmentation that sets the state of the art in visual object tracking. Its tracker, inherited from SAM 2, stores information from past frames in a memory bank, managed by a simple First-in-First-out rule, allowing temporal consistency in complex video sequences. Several recent methods improve SAM 2 with hand-crafted memory update rules, each targeting a specific failure mode, such as distractors, occlusions, or object motion. However, their gains transfer only partially to the stronger SAM 3 backbone. We instead present a more general, data-driven approach that frames memory control as a sequential decision-making task. SAM3RL is an extension of the SAM 3 tracker with a lightweight memory controller, trained by reinforcement learning, that decides which frames enter the memory bank in order to maximize tracking performance. SAM3RL sets a new state of the art on 10 of the 12 tracking and video object segmentation benchmarks, outperforming all hand-crafted rules without any runtime overhead. The same approach applied to SAM 2 gives SAM2RL, which outperforms hand-crafted rules on 9 of the 12 benchmarks, showing the generality of our method across model generations.

1 INTRODUCTION

Visual object tracking is a fundamental computer vision task in which a tracker is given a target object in the first frame of a video and must localize it, as a box or a segmentation mask, in all subsequent frames. Reliable tracking is a key component of applications such as autonomous driving, robotics, video surveillance, and sports analytics (Marvasti-Zadeh et al., 2022). The task remains difficult since the target may be surrounded by visually similar distractors, be occluded for long periods, or change its appearance. These challenges are amplified in long videos, where errors can accumulate over time and recovery depends critically on historical context (Javed et al., 2023).

Recent state-of-the-art trackers are built on the *Segment Anything Model* (SAM) family. SAM 2 (Ravi et al., 2025) is a promptable video segmentation model that conditions each mask prediction on features from past frames held in a small, fixed-size *memory bank*. SAM 3 (Carion et al., 2026) targets promptable concept segmentation, while inheriting the tracker architecture from SAM 2. The original SAM 2 populates the bank with a simple *First-In-First-Out* (FIFO) rule that keeps the most recent frames regardless of quality or relevance. Therefore, a line of follow-up work replaces FIFO with hand-crafted update rules, including motion-aware frame selection (Yang et al., 2026; Chen et al., 2026), distractor-aware memory (Videnović et al., 2026), and calibrated prototype memory with positional priors (Xu et al., 2025); some go further and also select among mask candidates by motion consistency (Yang et al., 2026; Chen et al., 2026) or maintain several banks and search over them (Ding et al., 2025b). SAM 3 also follows this approach and refines FIFO by keeping only frames in which the object is not occluded, plus the most recent frame.

However, each of the hand-crafted rules targets a particular failure mode, such as distractors, occlusions, or fast motion. Ideally, the tracker should be robust to all of them at the same time. Moreover, these rules are tied to the model generation they were designed for, as all heuristics developed for SAM 2 transfer poorly to the stronger SAM 3. Their gains on tracking benchmarks are limited, and on video object segmentation they even fall below SAM 3 on most benchmarks; see Figure 1.

In this work, we show that memory control should be learned rather than designed. We formulate memory control as a *Markov Decision Process* (MDP), designing the state, actions, and reward,

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

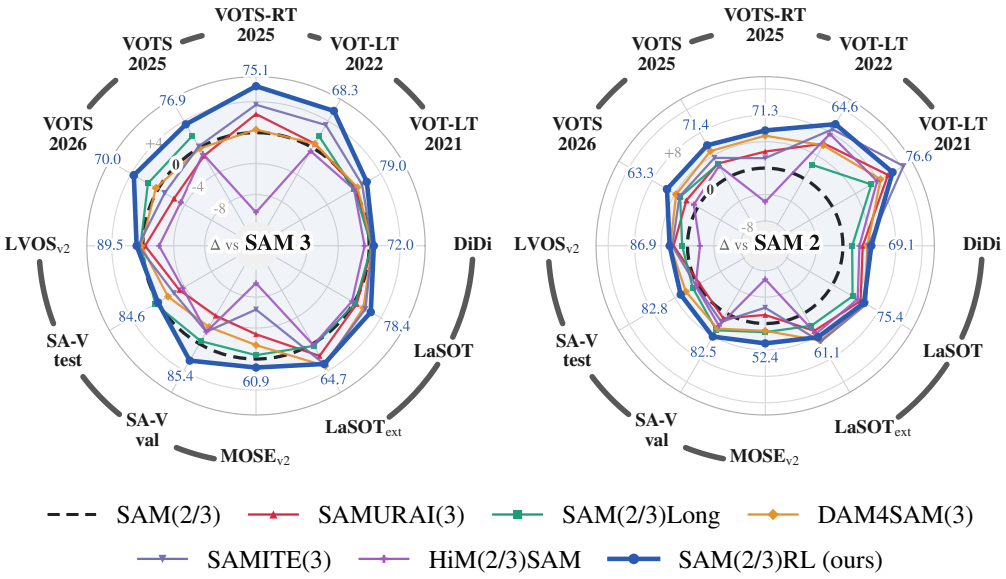

Figure 1: **Gains from hand-crafted rules vs. our learned memory control SAM3RL (left) and SAM2RL (right).** Each axis is a tracking or video segmentation benchmark, spanning three categories: VOT(S) challenges (VOTS$_{2025/2026}$, VOTS-RT$_{2025}$, and VOT-LT$_{2021/2022}$), visual object tracking (DiDi, LaSOT, LaSOT$_{ext}$), and video object segmentation (MOSE$_{v2}$, SA-V$_{val/test}$, LVOS$_{v2}$). The radius shows the change in that benchmark's standard metric relative to the corresponding baseline tracker (SAM 3 or SAM 2). The outer labels give our score on each benchmark. The compared methods modify what the tracker stores in the memory bank, and in most cases also mask selection or the number of banks. In contrast, our policy operates with the memory bank only. The compared methods are evaluated on SAM 2 as published and on SAM 3 after our porting to that backbone.

but leaving the choice of what to remember to the agent. That choice is learned by *reinforcement learning* (RL) directly from data, without touching the tracker itself. We introduce **SAM3RL**, which applies this memory control formulation to SAM 3 and sets a new state of the art on several tracking benchmarks, most notably on the VOTS and VOT-LT challenges. It does so at no additional inference cost and, unlike hand-crafted rules, without sacrificing performance on video object segmentation. Applied to SAM 2, the same recipe yields SAM2RL, which still outperforms hand-crafted rules on most benchmarks, showing the generality of our learned method across model generations.

Our contributions are as follows:

Learned memory control for SAM-based trackers. We formulate memory bank control in SAM-based trackers as an MDP over a fixed-capacity bank. The state describes the content of the bank, the action selects the entry to evict, and the reward scores the resulting prediction against the annotation. This formulation allows the memory control to be learned instead of being manually designed.

State-of-the-art visual object tracker. We present SAM3RL, an enhanced memory controller for SAM 3, obtained by optimizing the introduced MDP with RL. It improves over SAM 3 on 11 out of 12 benchmarks and ranks first on 10 of them. The largest gains are achieved on the VOT(S) challenges, with $+3.6$ on VOTS$_{2026}$ and $+5.5$ on VOT-LT$_{2022}$. It is also the only tracker that consistently succeeds in video object segmentation, scoring $+1.1$ on MOSE$_{v2}$ and $+2.5$ on SA-V$_{val}$. SAM3RL achieves this while optimizing the memory control only, whereas most compared methods additionally modify mask selection, prompting, or the number of banks maintained.

Online controller without a runtime penalty. The SAM3RL controller is a lightweight MLP evaluated once per memory update. It reads only values the tracker has already computed, and stays fully online without lookahead or extra feature extraction, unlike other methods. On tracking benchmarks, SAM3RL is faster than SAM 3, at 16.0 FPS against 15.3 FPS, as the memory is updated only every sixth frame, matching the decision rate used in training. Under the real-time FPS constraint of VOTS-RT$_{2025}$, SAM3RL stays best at 75.1, dropping only 1.8 points, while SAM 3 drops 4.3.

Generality across model generations. SAM2RL is obtained by applying the same recipe to SAM 2. It improves the baseline on all 12 benchmarks and leads on 9 of them, although every compared rule was designed specifically for SAM 2. SAM2RL still remains faster than SAM 2, at 25.5 FPS against 25.1 FPS. Together with SAM3RL, this demonstrates the generality of the introduced method.

2 Related Work

Visual object tracking (VOT) and *semi-supervised video object segmentation* (VOS) are conditioned on a target specified in the first frame of a video, which must then be localized in all subsequent frames. The target is represented by a bounding box in VOT and a pixel-level mask in VOS. Both tasks require maintaining the target through appearance changes, occlusions, fast motion, and distractors, which remain the dominant sources of tracking failure (Kristan et al., 2020). To address these challenges, trackers have long exploited information from previous frames (Matthews et al., 2004; Kalal et al., 2012; Oh et al., 2019; Cui et al., 2022; Han & Huang, 2024).

Recently, the SAM 2 foundation model (Ravi et al., 2025) set a new state of the art in tracking. Its impact is evident in the VOTS_{2026} challenge, where 92% of submitted trackers were derived from the SAM family (Kristan et al., 2026). Central to the SAM 2 design is the memory bank, a fixed-capacity store of past frames on which every prediction is conditioned by memory attention. The bank holds seven slots, the prompted first frame and the six most recent ones, under a FIFO rule. Because adjacent frames are highly correlated, FIFO tends to fill the bank with redundant observations rather than the diverse history required to distinguish distractors, or recover after occlusions.

Consequently, memory design in SAM-based trackers has become an active research area, with follow-up work replacing FIFO with hand-crafted selection rules. SAMURAI (Yang et al., 2026) uses a Kalman filter (Kalman, 1960) to score motion consistency, both selecting among per-frame mask candidates and gating memory updates. DAM4SAM (Videnović et al., 2026) splits the bank into a fixed-interval part and a part reserved for frames that resolve visually similar distractors. SAMITE (Xu et al., 2025) calibrates stored prototypes and adds positional priors. HiM2SAM (Chen et al., 2026) combines hierarchical motion estimation with a long- and short-term split of the bank. SAM2Long (Ding et al., 2025b) keeps several banks in parallel and performs tree search over alternative trajectories. Even SAM 3 (Carion et al., 2026) relies on such a rule, retaining an unprompted frame only when the model is confident that the target is present.

However, for older trackers, learned memory control has already been shown to outperform hand-crafted rules (Zhang et al., 2019b; Yang & Chan, 2018; Dai et al., 2020; Choi et al., 2018). Specifically for SAM 2, learning the rules instead of designing them was attempted by Adamyan et al. (2025). They showed the potential of learned memory control by fitting a separate RL agent to each of a handful of training videos. Agents improve over the hand-crafted rules, but do not transfer to unseen videos. This demonstrates a performance upper bound rather than providing a usable tracker. In contrast, we develop the full method, a single general-purpose policy that generalizes to unseen videos, transfers to SAM 3, and sets a new state of the art in visual object tracking.

3 Methodology

We formulate memory control as an MDP and train a lightweight MLP controller π_θ with RL by rolling out tracking episodes over a corpus of annotated videos. Given a training sequence with frames $x_0, \ldots, x_T$ and object mask or box annotations $y_0, \ldots, y_T$, the simulation starts by conditioning the frozen tracker on the first annotation y_0; see Figure 2. At each step t, the tracker holds a memory bank $B_t = (m_{\tau_0}, m_{\tau_1}, \ldots, m_{\tau_6})$ of seven memory frames with indices $0 = \tau_0 < \tau_1 < \cdots < \tau_6 < t$, where slot 0 is permanently occupied by the conditioning frame m_0 and the other six hold past memory frames formed from the tracker's own predictions. The image encoder embeds x_t and the memory attention conditions this embedding on the current content of B_t. The mask decoder then converts the result into three candidate masks and keeps the most confident as the prediction $\hat{y}_t$. The mask decoder also outputs an object pointer p_t, which is a single token summarizing the appearance of the target. The memory encoder then fuses $\hat{y}_t$ with the embedding of x_t into a spatial memory feature f_t, and the two together form the memory frame $m_t = (p_t, f_t)$.

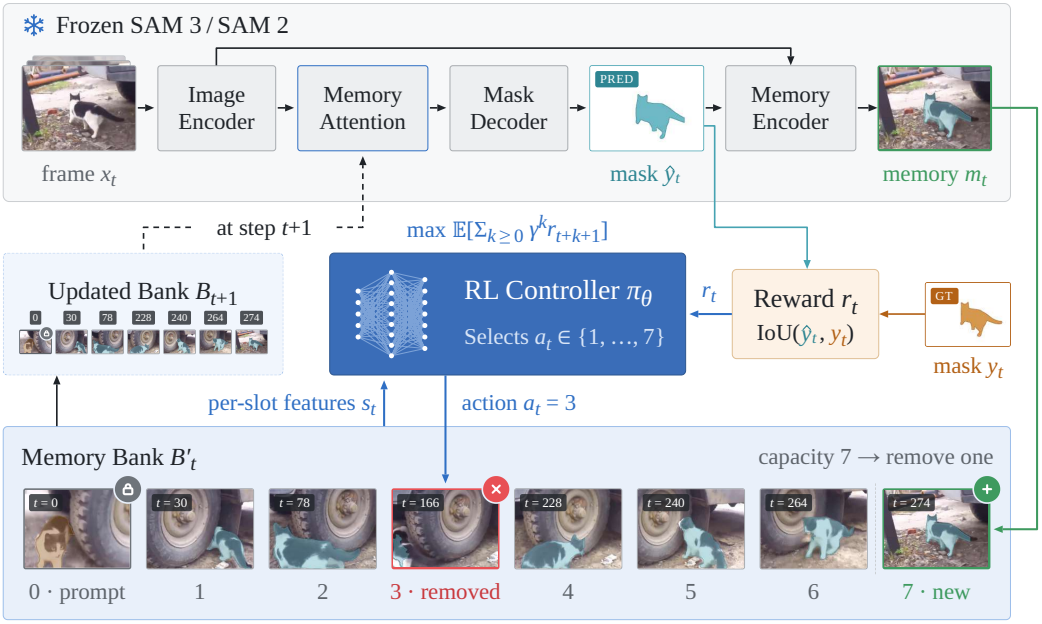

Figure 2: **Learned memory control for SAM-based trackers.** The frozen tracker (top) embeds the frame x_t and conditions it through memory attention on the current memory bank B_t. The decoder then predicts the mask $\hat{y}_t$ and fuses it with the embedded x_t into a new memory frame m_t. The bank has capacity seven, where slot 0 is permanently reserved for the conditioning frame. Appending m_t to slot 7 therefore overfills the bank, so one of the slots $1, \ldots, 7$ must be evicted. Our controller π_θ observes the state s_t, which collects temporal, confidence, semantic, and geometric features of all eight slots, and picks the action $a_t \in \{1, \ldots, 7\}$ that removes the corresponding memory (here $a_t = 3$). The surviving slots form the updated bank B_{t+1} that the tracker reads at step $t+1$. During training, π_θ receives the reward r_t, scoring $\hat{y}_t$ against the annotation y_t, and maximizes the expected discounted return $\mathbb{E}_{\pi_\theta}\big[\sum_{k\geq 0}\gamma^k r_{t+k+1}\big]$. At inference, π_θ is fixed and the actions are taken greedily.

We assume that the bank B_t is always full, except during the first six frames $x_1, \ldots, x_6$, when the memory frames $m_1, \ldots, m_6$ are simply appended without eviction. Once B_t is filled, storing m_t requires evicting one of the existing entries. Whereas prior work relies on hand-crafted rules, we employ a learned controller π_θ that selects one of seven candidates for removal, either one of the stored memories $m_{\tau_1}, \ldots, m_{\tau_6}$ or the newly formed memory m_t. The six remaining candidates, together with the conditioning memory m_0, form the updated bank B_{t+1} used for the tracker prediction in the next step. Once training is complete, the controller π_θ is fixed and the same loop is used for inference on unseen videos, with actions taken greedily and no reward computed.

3.1 MDP Formulation

An *episode* of our MDP is one object tracked through a video. The policy π_θ acts once per tracker step, after $\hat{y}_t$ and m_t have been produced and before B_t is updated. We denote a temporary bank after appending the new memory frame m_t by $B'_t = (m_{\tau_0}, \ldots, m_{\tau_7})$ where $\tau_7 = t$. The policy observes the state s_t and selects the action a_t that removes one of $m_{\tau_1}, \ldots, m_{\tau_7}$, producing the next memory bank B_{t+1}. It also receives the reward r_t, which reflects the quality of the prediction $\hat{y}_t$ against the annotation y_t. We now formalize the state s_t, action a_t, and reward r_t.

State space. The state s_t describes the content of B'_t by four pairwise disjoint groups of features, computed for every slot $i \in \{0, \ldots, 7\}$ from the tracker outputs at the corresponding time step τ_i:

- *Temporal encodings* are sinusoidal positional encodings $\mathrm{PE}_{\tau_i} \in \mathbb{R}^{32}$ of the frame age $t - \tau_i$.

- *Confidence features* are the object score $o_{\tau_i} \in \mathbb{R}$, which is the logit of the target being visible in x_{τ_i}, and the predicted IoU scores $\iota_{\tau_i} \in [0,1]^3$, one per candidate mask, estimating how well that mask overlaps with the ground truth. Both o_{τ_i} and ι_{τ_i} are outputs of the tracker's mask decoder.
- *Semantic features* are the object pointer $p_{\tau_i} \in \mathbb{R}^{256}$ and the mean-pooled memory feature $f^{pool}_{\tau_i} \in \mathbb{R}^{64}$ of the spatial memory feature f_{τ_i} under the predicted mask $\hat{y}_{\tau_i}$ to reduce the input size. The features also contain pairwise cosine similarities between (p_{τ_i}, p_{τ_j}) and $(f^{pool}_{\tau_i}, f^{pool}_{\tau_j})$ for $i \neq j$.
- *Geometric features* describe the area, centroid, and box size of $\hat{y}_{\tau_i}$, plus the pairwise mask IoUs between $\hat{y}_{\tau_i}$ and $\hat{y}_{\tau_j}$ for $i \neq j$. They further contain the IoUs of the two unselected candidate masks for $\hat{y}_t$ against every $\hat{y}_{\tau_i}$, as well as the IoU between the two unselected masks themselves.

Action space. The action $a_t \in \{1, \ldots, 7\}$ is the index of the memory frame that the policy removes from B'_t, giving the updated bank $B_{t+1} = B'_t \setminus \{m_{\tau_{a_t}}\}$. Choosing $a_t = 7$ removes the newly formed m_t and thus rejects the current frame, leaving the bank unchanged at $B_{t+1} = B_t$. The actions also determine which object pointers the tracker attends to. In fact, memory attention reads sixteen object pointers while only seven memory features. We take the seven memory features and object pointers from B_t and fill the remaining nine pointers from the frames most recently evicted from the bank. We keep temporal encodings for the memory attention inputs from SAM 3 but index object pointers by recency rank instead of frame offset, so their encodings stay within the range seen in training.

The policy then defines a distribution $\pi_\theta(a_t \mid s_t)$ over the seven actions, from which we sample during training and take the most likely action during inference. The memory rules of SAM 2 and SAM 3 can themselves be formulated as policies in this action space. SAM 2 always selects $a_t = 1$ and evicts the oldest memory, while SAM 3 keeps the most recent frame and selects $a_t = 6$ whenever an object-score-gated IoU of x_{τ_6} falls below a predefined threshold, and $a_t = 1$ otherwise.

Reward. The reward measures the tracking quality of the prediction $\hat{y}_t$ against the annotation y_t by

$$q_t = \begin{cases} 1 & \text{if } \hat{y}_t = \emptyset \text{ and } y_t = \emptyset, \\ \mathrm{IoU}(\hat{y}_t, y_t) & \text{otherwise,} \end{cases} \tag{1}$$

which credits the tracker for the overlap between $\hat{y}_t$ and y_t when the target is visible, and for correctly reporting when the object is absent. If the annotation is a bounding box, the IoU is computed against the tight axis-aligned box enclosing the predicted mask. A large amount of variance of q_t comes from the difficulty of the video rather than from the actions taken. Therefore, we normalize the reward by setting $r_t = q_t - b_t$, where b_t is the tracking quality that the underlying tracker scores at the same frame of the same video under the same conditioning with SAM 3's default memory control. The baseline depends only on the video and not on the policy π_θ, so we pre-compute b_t only once for the whole training dataset. The policy then maximizes the expected return $\mathbb{E}_{\pi_\theta}[G_t]$, where $G_t = \sum_{k \geq 0} \gamma^k r_{t+k+1}$ is the discounted return from step t and γ is the discount factor.

Appendix E ablates the input feature groups, object pointer selection, and the reward normalization.

3.2 Training the Memory Controller

Architecture. The controller π_θ is a single MLP, which serves as the *actor*. All features enter it directly, except the object pointers, which a shared linear projection first compresses to 64 dimensions. All groups are then concatenated and passed through one hidden layer of 2,048 units and a linear head over the seven actions, about 3.2 M parameters in total. For training, we add a *critic*, a second network of the same design with a scalar head, which estimates the expected return, shares no parameters with the actor, and is discarded after training. Besides s_t, the critic receives the normalized number of decision steps left in the episode, which it needs to predict undiscounted finite-horizon returns. This value is unavailable at inference, but since only the actor is deployed, the controller remains fully online and depends on s_t alone.

Training data. We train on the LaSOT training split (Fan et al., 2019), whose 1,120 sequences are divided into 896 for training and 224 for validation, while the official test set is held out. LaSOT is annotated with per-frame bounding boxes at 30 FPS. Since consecutive frames at that rate are nearly identical, an eviction decision changes the bank very little while still consuming a decision step. We therefore subsample every sixth frame, so that training episodes run at 5 FPS and span 425 decision steps on average. This also matches the typical annotation rate of VOS datasets.

Optimization. We train the controller with Proximal Policy Optimization (PPO) (Schulman et al., 2017), using the undiscounted Monte-Carlo return (i.e., $\gamma = 1$) minus the critic's predicted value as the advantage. The return is bootstrapped only when the rollout window truncates an episode, corresponding to $\lambda = 1$ in generalized advantage estimation (GAE) (Schulman et al., 2016). Apart from this bootstrap, we keep the critic out of the targets and use it only as a baseline, as it fits the training videos well but transfers poorly to unseen videos. Each of the 400 iterations collects 376-step rollouts from 256 parallel environments. Training takes about 24 hours on four H200 GPUs. We evaluate the greedy policy on the full validation videos every 25 iterations and select the best checkpoint. All remaining hyperparameters are listed in Table 13 in Appendix D.

Inference. The actor picks actions greedily, and its only overhead is a single MLP evaluation per memory update. To keep the decision rate close to the 5 FPS seen in training, the memory controller operates with a stride $s = \text{round}(\nu/5)$, where ν is the frame rate of the benchmark. The tracker still predicts a mask on every frame, but only every sth frame triggers a memory update and can enter the bank. On the remaining frames no memory frame m_t is formed, so the memory encoder that produces f_t is skipped altogether. The 30 FPS VOT and VOTS benchmarks use $s = 6$, while the 5 FPS MOSE_{v2} uses $s = 1$. The frame rates and strides for all benchmarks are listed in Appendix B.

4 SAM3RL Performance

SAM3RL sets a new state of the art among memory control methods for SAM-based trackers on most tracking and video object segmentation benchmarks. In this section, we compare it with the prior methods discussed in Section 2, all evaluated on the SAM 3 backbone. Because these methods were designed for SAM 2 and their authors have not released SAM 3 implementations, we port each one to SAM 3 ourselves. Each port replaces the SAM 2 video predictor with the SAM 3 tracker and leaves the method otherwise untouched. We also compare the prior methods, in their original SAM 2 implementations, against SAM2RL (our memory control on SAM 2) in Figure 1 and Appendix A. Appendix B details tracker evaluation, FPS measurement, reproducibility, and the ports.

4.1 VOT(S) Challenges

The VOT initiative has organized annual tracking challenges for over a decade, and its benchmarks have evolved with the field (Kristan et al., 2013). Current VOTS challenges unify short-term and long-term tracking with single-target and multi-target scenarios. The sequences feature occlusions, visually similar distractors, and targets that leave the field of view and re-enter. VOTS targets are annotated by segmentation masks and scored on a remote server, whereas VOT-LT targets are annotated by bounding boxes. Results are presented in Table 1. The challenges separate the methods most sharply, with scores spanning 7.0 points on VOTS_{2026}. SAM3RL ranks first on all five challenges. Among hand-crafted rules, SAMITE3 and SAM3Long provide the strongest baselines.

VOTS_{2026} (Kristan et al., 2026) is the latest edition of the challenge, and **VOTS_{2025}** (Kristan et al., 2023b) is the previous one, whose sequences are a subset of it. **VOTS-RT_{2025}** (Kristan et al., 2025) is the real-time variant of the VOTS_{2025} benchmark, designed to evaluate tracking quality under a strict streaming constraint. Frames are delivered to the tracker at a constant rate of 20 FPS, independently of its processing latency. The tracker must finish reporting the mask in the current frame before the next frame becomes available. Otherwise, the prediction from the previous frame is retained. The constraint costs SAM 3 several points, whereas SAM3RL gains the most of all methods. Notably, we do not report SAM3Long in this setting because it is not an online tracker and cannot satisfy this causal protocol; see Appendix B. **VOT-LT_{2021}** (Kristan et al., 2021) evaluates long-term tracking, where the target is absent for extended periods and must be redetected, and **VOT-LT_{2022}** (Kristan et al., 2023a) is the latest LT challenge, which introduced more difficult sequences. The results on VOT-LT_{2021} were the most affected by a race condition in the original SAM 3 video predictor, which we identify and fix in Appendix C; the numbers reported in this work are measured after the fix.

Model throughput is measured on VOTS_{2026} and covers model inference only. On the 30 FPS benchmarks, SAM3RL updates its memory only every sth frame with $s = 6$, matching the 5 FPS data used during training. On the $s - 1$ frames in between, no memory frame is formed, so the memory encoder is skipped and the controller is not evaluated. SAM 3, in contrast, runs the memory encoder on every frame. The time saved on memory encoding outweighs the cost of the controller,

Table 1: **Performance on VOT(S) challenges.** We report standard metrics for the evaluated challenges. Deltas indicate improvement or decline relative to SAM 3. **Bold** indicates the best.

		Quality			F-score	
Model	FPS	VOTS_{2026}	VOTS_{2025}	VOTS-RT_{2025}	VOT-LT_{2021}	VOT-LT_{2022}
SAM 3	15.3	66.4	73.4	69.1	77.1	62.8
SAMURAI3	15.1	64.0 -2.4	72.2 -1.2	71.5 +2.4	77.4 +0.3	63.4 +0.6
SAM3Long	9.3	67.9 +1.5	75.2 +1.8	–	77.1 ±0.0	64.5 +1.7
DAM4SAM3	14.2	66.7 +0.3	73.3 -0.1	69.5 +0.4	77.6 +0.5	63.4 +0.6
SAMITE3	12.7	65.4 -1.0	73.6 +0.2	72.7 +3.6	78.3 +1.2	66.2 +3.4
HiM3SAM	11.8	63.0 -3.4	72.5 -0.9	58.8 -10.3	77.1 ±0.0	62.3 -0.5
SAM3RL (ours)	**16.0**	**70.0** +3.6	**76.9** +3.5	**75.1** +6.0	**79.0** +1.9	**68.3** +5.5

Table 2: **Performance on VOT datasets.** Deltas indicate improvement or decline relative to SAM 3. **Bold** indicates the best.

	Quality	AUC	
Model	DiDi	LaSOT	$\text{LaSOT}_{\text{ext}}$
SAM 3	71.4	75.9	61.7
SAMURAI3	71.8 +0.4	76.3 +0.4	63.5 +1.8
SAM3Long	71.7 +0.3	76.1 +0.2	62.0 +0.3
DAM4SAM3	71.8 +0.4	77.3 +1.4	**65.0** +3.3
SAMITE3	71.8 +0.4	77.6 +1.7	64.7 +3.0
HiM3SAM	70.8 -0.6	75.6 -0.3	61.9 +0.2
SAM3RL	**72.0** +0.6	**78.4** +2.5	64.7 +3.0

Table 3: **Performance on VOS datasets.** Deltas indicate improvement or decline relative to SAM 3. **Bold** indicates the best.

	$\mathcal{J}\&\dot{\mathcal{F}}$	$\mathcal{J}\&\mathcal{F}$		
Model	MOSE_{v2}	SA-V_{val}	$\text{SA-V}_{\text{test}}$	LVOS_{v2}
SAM 3	59.8	82.9	84.6	88.7
SAMURAI3	56.6 -3.2	78.7 -4.2	81.4 -3.2	88.4 -0.3
SAM3Long	59.3 -0.5	82.5 -0.4	**85.0** +0.4	89.0 +0.3
DAM4SAM3	58.0 -1.8	80.4 -2.5	83.1 -1.5	88.9 +0.2
SAMITE3	53.4 -6.4	81.1 -1.8	82.2 -2.4	89.3 +0.6
HiM3SAM	50.0 -9.8	81.1 -1.8	80.9 -3.7	86.6 -2.1
SAM3RL	**60.9** +1.1	**85.4** +2.5	84.6 ±0.0	**89.5** +0.8

so SAM3RL reaches 16.0 FPS, exceeding both the hand-crafted rules and the 15.3 FPS of SAM 3. The speedup therefore comes from the less frequent memory updates rather than from the policy itself. With memory updated on every frame where $s = 1$, SAM3RL runs at 14.9 FPS; see Table 4.

4.2 VOT Datasets

The VOT datasets are annotated with bounding boxes, so each predicted mask is scored by its tight box; results are in Table 2. SAM3RL improves over the SAM 3 on all three and ranks first on two of them. DAM4SAM3 and SAMITE3 are the strongest rules here, while HiM3SAM is the weakest.

DiDi (Videnović et al., 2026) is a set of distractor-distilled sequences from various tracking datasets, built to isolate the failures caused by visually similar objects. It was introduced with DAM4SAM, which was designed specifically against distractors. On SAM 2, DAM4SAM improves clearly on DiDi (Appendix A), but on SAM 3 most of the gain disappears. **LaSOT** (Fan et al., 2019) is the standard long-term tracking benchmark. Our policy is trained on the LaSOT training split, so its test set is the closest to our training distribution among all benchmarks. **$\text{LaSOT}_{\text{ext}}$** (Fan et al., 2021) extends LaSOT with sequences of object classes not present in the LaSOT training set. It is the only tracking benchmark in the main comparison on which SAM3RL does not rank first. The gap to the best method is small, and the three leading methods stay well ahead of the remaining rules.

4.3 VOS Datasets

The VOS datasets are annotated with segmentation masks, so predictions are scored at the pixel level; results are in Table 3. SAM3RL improves over the baseline on three of the four benchmarks and ranks first on them. The rules that help most on tracking hurt segmentation, whereas our learned memory controller, trained only on box-annotated LaSOT, remains effective in both settings.

Table 4: **Memory stride ablation.** All benchmarks here run at 30 FPS, so $s = 6$ matches the 5 FPS training and $s = 1$ updates every frame. SAM 3 at $s = 1$ is the original tracker. SAM 3 at $s = 6$ is the official strided variant of SAM 3, which encodes memory and reselects the bank every frame t, keeping $t-1$ frame plus the frames $t-1-s, t-1-2s, \ldots$ that pass SAM 3's threshold. SAM3RL and SAM 3_{ours} update the bank only every sth frame and hold it fixed in between, with SAM 3_{ours} using SAM 3's selection rule. Deltas are relative to SAM 3 at $s = 1$. **Bold** indicates the best.

			Quality			F-score		AUC	
Model	s	FPS	VOTS_{2026}	VOTS_{2025}	DiDi	VOT-LT_{2021}	VOT-LT_{2022}	LaSOT	$\text{LaSOT}_{\text{ext}}$
SAM 3	1	15.3	66.4	73.4	71.4	77.1	62.8	75.9	61.7
SAM3RL	1	14.9	69.3 +2.9	75.4 +2.0	71.7 +0.3	77.2 +0.1	65.6 +2.8	77.1 +1.2	**65.2** +3.5
SAM 3	6	15.5	55.5 -10.9	66.9 -6.5	60.5 -10.9	76.2 -0.9	61.8 -1.0	72.0 -3.9	57.4 -4.3
SAM 3_{ours}	6	15.8	68.1 +1.7	75.9 +2.5	71.2 -0.2	75.7 -1.4	65.6 +2.8	77.4 +1.5	64.8 +3.1
SAM3RL	6	**16.0**	**70.0** +3.6	**76.9** +3.5	**72.0** +0.6	**79.0** +1.9	**68.3** +5.5	**78.4** +2.5	64.7 +3.0

MOSE_{v2} (Ding et al., 2025a) targets conditions largely absent from the classic VOS benchmarks, including frequent disappearance and reappearance, severe occlusions and crowding, small objects, and adverse lighting. It is the hardest VOS benchmark in our evaluation. All hand-crafted rules fall below the baseline, and SAM3RL is the only method to improve on it. **LVOS_{v2}** (Hong et al., 2024) evaluates long-term segmentation, where the memory bank must retain information over far longer horizons than on the other VOS benchmarks. It is the only VOS benchmark on which most rules improve over the baseline. **SA-V** (Ravi et al., 2025) was released with SAM 2, covering diverse everyday objects. SAM3RL leads on the validation split and matches the baseline on the test split.

5 Analysis of Learned Memory Control

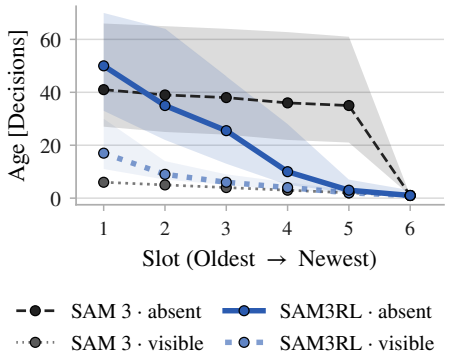

Figure 3: **Frame age in memory slots** for SAM3RL and SAM 3 on MOSEv2. *Visible* pools all decisions with the target present. *Absent* pools the last decisions of absent gaps of at least 16 decisions. Median with the 25th to 75th percentile.

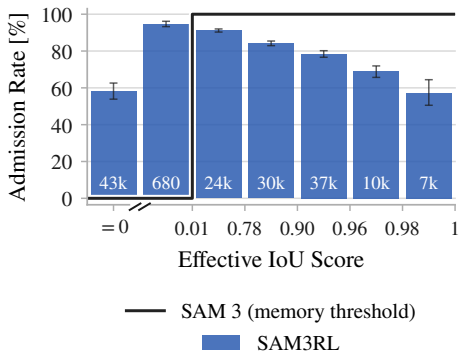

Figure 4: **Admission rate of incoming frames** by SAM3RL on MOSEv2 against the frame's predicted IoU score scaled by $\max(2\sigma(o_t) - 1, 0)$. SAM 3 stores only frames above 0.01. Decisions are binned, counts in white, 95% CIs.

Memory stride. Deciding on every frame is the most flexible choice in principle, but training at 30 FPS is costly, as most decisions concern nearly identical frames. We therefore train at 5 FPS and match that decision rate at inference with the stride s. Table 4 shows the effect of stride on 30 FPS benchmarks. At $s = 1$, our policy runs outside its training distribution, yet still outperforms SAM 3 on all seven benchmarks, by 1.8 points on average. At $s = 6$, although only every sixth frame can enter the bank, the gain grows to 2.9 points as the decision rate matches the 5 FPS used in training. In our evaluations, the compared methods run at their native strides, as most were developed for 30 FPS benchmarks, and DAM4SAM and HiM2SAM even adapt their stride to the benchmark frame rate, similarly to ours. The SAM 3 paper, however, reports only VOS results at $s = 1$, so we also evaluate SAM 3 at $s = 6$, both with its official strided variant and as our modification SAM 3_{ours}, which pairs

SAM 3's selection rule with our stride. The former loses up to 10.9 points, while the latter gains 1.4 points on average. At equal stride, SAM3RL still outperforms SAM 3_{ours} by 1.5 points on average.

Learned memory selection. SAM 3 admits only frames whose effective IoU score exceeds a threshold. Its bank thus acts as FIFO while the target is visible and, during absences, holds the pre-absence frames plus the most recent one. Figure 3 shows that SAM3RL instead spreads its bank further back in time and, during absences, mixes older pre-absence frames with ones from the absence itself. Unlike SAM 3, the policy admits nearly all medium-score frames and grows more selective as the score rises, rejecting the most confident frames about as often as those predicted absent; see Figure 4. The policy also adapts to video content, admitting far more frames when the target moves or changes appearance, as in Figure 5. Appendix F quantifies this effect and includes more behavior analysis.

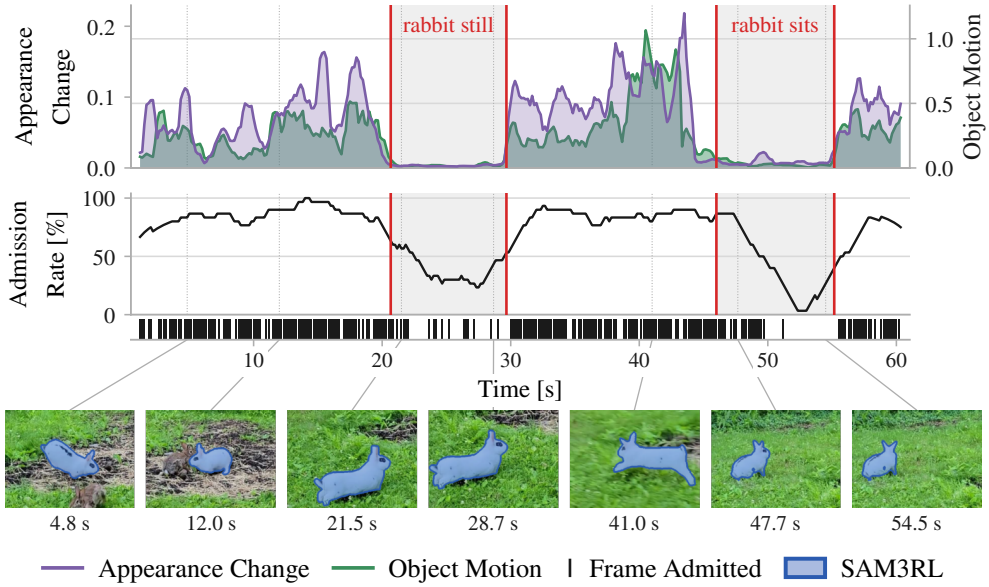

Figure 5: **Admission rate of incoming frames by SAM3RL follows object change.** An LVOS_{v2} video sequence in which a rabbit settles and hops off twice over 60 s. *Object motion* is the shift of the ground-truth box center between consecutive decisions, normalized by the square root of the box area. *Appearance change* is the DINOv3 (Siméoni et al., 2025) cosine distance between consecutive ground-truth crops. The middle panel is the admission rate in a 5 s window, one tick per admitted decision. Admissions are sparse in the sitting intervals (red) and dense when both curves rise.

6 Discussion

We presented SAM3RL, which learns memory control over a fixed-capacity bank with RL while keeping the backbone frozen. SAM3RL improves over SAM 3 on 11 of 12 benchmarks and ranks first on 10, with the largest gains on VOT(S), at virtually no runtime cost. It is also the only method that consistently improves on video object segmentation. Applying the same recipe to SAM 2 yields SAM2RL, which improves over SAM 2 on every benchmark and leads on 9. These results suggest that memory control should be learned rather than designed, as learned control improves even the strongest trackers and, unlike fixed rules, remains effective across model generations.

Limitations. First, the policy is tied to the 5 FPS decision rate seen in training. Deciding on every frame of 30 FPS benchmarks still improves over SAM 3, but the average gain drops from 2.9 to 1.8 points. We therefore match the training rate with the stride s, which requires the frame rate to be known in advance and keeps the skipped frames out of the memory bank. The former is rarely restrictive in practice, as the frame rate is usually available. Training at the native rate would remove this restriction, but most decisions would then concern near-identical frames. Such a policy could nonetheless exploit the skipped frames and improve further on high-frame-rate benchmarks. Second, the policy controls only the seven frames in the memory bank, while the nine object pointers are filled by a fixed rule. Learning both jointly is a natural direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, implement methods, assist with translation, support qualitative and thematic data analysis, and interpret results. We have not used generative AI tools to help develop theoretical models or conceptual frameworks. The following are not applicable to this work: generate synthetic data sets, clean and reformat dataset, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, and propose or refine hypotheses. Additionally, we used generative AI tools to edit a research paper to improve readability, create or modify scientific figures or images, creation of artifacts, draft parts of a research paper, brainstorming, suggest experimental parameters, create or edit software code, identify relevant literature, sourcing/searching for information, and summarize or analyze existing literature. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. LLM-generated code was verified and tested for correctness. We checked the correctness of the numbers in the LLM-generated visualizations. We checked all the LLM-visual refinements of the text, tables, and figures. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the full source code as supplementary material. It includes the environment, the policy, and the training script of SAM3RL and SAM2RL, our ports of the compared methods to SAM 3, the original SAM 2 implementations of these methods as evaluated, the evaluation scripts for every benchmark in the paper, and the initialisation masks used for the tracking benchmarks. The MDP, the policy architecture, and the training procedure are described in Section 3. The complete list of hyperparameters, the training data, and the checkpoint selection rule are given in Appendix D. Every benchmark we evaluate on is listed in Table 8 with its number of sequences, objects, frame rate, and length. The evaluation protocols, the metrics, and the toolkits used to score each benchmark, together with the reproduction of published baseline numbers, are described in Appendix B. Appendix C documents a race condition in the SAM 3 implementation and the fix we apply, which affects the reported numbers of all methods evaluated on that backbone.

REFERENCES

Alen Adamyan, Tomáš Čížek, Matej Straka, Klára Janoušková, and Martin Schmid. SAM2RL: Towards Reinforcement Learning Memory Control in Segment Anything Model 2, 2025. URL https://arxiv.org/abs/2507.08548.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, and Ronghang Hu et al. SAM 3: Segment Anything with Concepts. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 138846–138923, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e0982cbc81401df3430ee1ff780dc7a2-Paper-Conference.pdf.

Ruixiang Chen, Guolei Sun, Yawei Li, Jie Qin, and Luca Benini. HiM2SAM: Enhancing SAM2 with Hierarchical Motion Estimation and Memory Optimization towards Long-term Tracking. In Josef Kittler, Hongkai Xiong, Jian Yang, Xilin Chen, Jiwen Lu, Weiyao Lin, Jingyi Yu, and Weishi Zheng (eds.), *Pattern Recognition and Computer Vision*, pp. 276–291, Singapore, 2026. Springer Nature Singapore. ISBN 978-981-95-5755-4.

Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Real-time visual tracking by deep reinforced decision making. *Computer Vision and Image Understanding*, 171:10–19, 2018. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2018.05.009. URL https://www.sciencedirect.com/science/article/pii/S1077314218300754.

Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. MixFormer: End-to-End Tracking with Iterative Mixed Attention. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13598–13608, 2022. doi: 10.1109/CVPR52688.2022.01324.

Kenan Dai, Yunhua Zhang, Dong Wang, Jianhua Li, Huchuan Lu, and Xiaoyun Yang. High-Performance Long-Term Tracking With Meta-Updater. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6297–6306, 2020. doi: 10.1109/CVPR42600.2020.00633.

Martin Danelljan, Goutam Bhat, Matteo Dunnhofer, and Christoph Mayer. PyTracking: Visual tracking library based on PyTorch. `https://github.com/visionml/pytracking`, 2019. Accessed September 2026.

Henghui Ding, Kaining Ying, Chang Liu, Shuting He, Xudong Jiang, Yu-Gang Jiang, Philip H. S. Torr, and Song Bai. MOSEv2: A More Challenging Dataset for Video Object Segmentation in Complex Scenes, 2025a. URL `https://arxiv.org/abs/2508.05630`.

Shuangrui Ding, Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Yuwei Guo, Dahua Lin, and Jiaqi Wang. SAM2Long: Enhancing SAM 2 for Long Video Segmentation with a Training-Free Memory Tree. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13614–13624, October 2025b.

Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Harshit, Mingzhen Huang, Juehuan Liu, Yong Xu, Chunyuan Liao, Lin Yuan, and Haibin Ling. LaSOT: A High-quality Large-scale Single Object Tracking Benchmark. *International Journal of Computer Vision*, 129(2):439–461, 2021. ISSN 1573-1405. doi: 10.1007/s11263-020-01387-y. URL `https://doi.org/10.1007/s11263-020-01387-y`.

Yushan Han and Kaer Huang. ACTrack: Adding Spatio-Temporal Condition for Visual Object Tracking, 2024. URL `https://arxiv.org/abs/2403.07914`.

Lingyi Hong, Zhongying Liu, Wenchao Chen, Chenzhi Tan, Yuang Feng, Xinyu Zhou, Pinxue Guo, Jinglun Li, Zhaoyu Chen, Shuyong Gao, Wei Zhang, and Wenqiang Zhang. Lvos: A benchmark for large-scale long-term video object segmentation, 2024. URL `https://arxiv.org/abs/2404.19326`.

Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1562–1577, 2021. doi: 10.1109/TPAMI.2019.2957464.

Sajid Javed, Martin Danelljan, Fahad Shahbaz Khan, Muhammad Haris Khan, Michael Felsberg, and Jiri Matas. Visual Object Tracking With Discriminative Filters and Siamese Networks: A Survey and Outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6552–6574, 2023. doi: 10.1109/TPAMI.2022.3212594.

Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. doi: 10.1109/TPAMI.2011.239.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL `https://doi.org/10.1115/1.3662552`.

Matej Kristan, Roman Pflugfelder, Ales Leonardis, and Jiri Matas et al. The Visual Object Tracking VOT2013 Challenge Results. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, June 2013.

Matej Kristan, Aleš Leonardis, and Jiří Matas et al. The Eighth Visual Object Tracking VOT2020 Challenge Results. In Adrien Bartoli and Andrea Fusiello (eds.), *Computer Vision – ECCV 2020 Workshops*, pp. 547–601, Cham, 2020. Springer International Publishing. ISBN 978-3-030-68238-5.

Matej Kristan, Jiří Matas, and Aleš Leonardis et al. The Ninth Visual Object Tracking VOT2021 Challenge Results. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 2711–2738, 2021. doi: 10.1109/ICCVW54120.2021.00305.

Matej Kristan, Aleš Leonardis, and Jiří Matas et al. The Tenth Visual Object Tracking VOT2022 Challenge Results. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino (eds.), *Computer Vision – ECCV 2022 Workshops*, pp. 431–460, Cham, 2023a. Springer Nature Switzerland. ISBN 978-3-031-25085-9.

Matej Kristan, Jiří Matas, and Martin Danelljan et al. The First Visual Object Tracking Segmentation VOTS2023 Challenge Results. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 1788–1810, 2023b. doi: 10.1109/ICCVW60793.2023.00195.

Matej Kristan, Jiří Matas, Pavel Tokmakov, and Alan Lukežič et al. The Third Visual Object Tracking Segmentation VOTS2025 Challenge Results. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 7472–7490, 2025. doi: 10.1109/ICCVW69036.2025.00771.

Matej Kristan, Jiri Matas, Pavel Tokmakov, and Adam W Harley et al. The Fourth Visual Object Tracking Segmentation VOTS2026 Challenge Results, 2026. URL https://openreview.net/forum?id=zGOan36xpF.

Alan Lukeźič, Luka Čehovin Zajc, Tomáš Vojíř, Jiří Matas, and Matej Kristan. Performance Evaluation Methodology for Long-Term Single-Object Tracking. *IEEE Transactions on Cybernetics*, 51(12):6305–6318, 2021. doi: 10.1109/TCYB.2020.2980618.

Seyed Mojtaba Marvasti-Zadeh, Li Cheng, Hossein Ghanei-Yakhdan, and Shohreh Kasaei. Deep Learning for Visual Tracking: A Comprehensive Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(5):3943–3968, May 2022. ISSN 1558-0016. doi: 10.1109/tits.2020.3046478. URL http://dx.doi.org/10.1109/TITS.2020.3046478.

L. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004. doi: 10.1109/TPAMI.2004.16.

Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video Object Segmentation Using Space-Time Memory Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 724–732, 2016. doi: 10.1109/CVPR.2016.85.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation, 2018. URL https://arxiv.org/abs/1704.00675.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, and Tengyu Ma et al. SAM 2: Segment Anything in Images and Videos. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/45c1f6a8cbf2da59ebf2c802b4f742cd-Paper-Conference.pdf.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*, 2016. URL https://mlanthology.org/iclr/2016/schulman2016iclr-high/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, and Federico Baldassarre et al. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

Jovana Videnović, Matej Kristan, and Alan Lukežič. Distractor-Aware Memory-Based Visual Object Tracking. In *International Journal of Computer Vision*, 2026.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object Tracking Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. doi: 10.1109/TPAMI.2014.2388226.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Qianxiong Xu, Lanyun Zhu, Chenxi Liu, Guosheng Lin, Cheng Long, Ziyue Li, and Rui Zhao. SAMITE: Position Prompted SAM2 with Calibrated Memory for Visual Object Tracking, 2025. URL https://arxiv.org/abs/2507.21732.

Cheng-Yen Yang, Hsiang-Wei Huang, Wenhao Chai, Zhongyu Jiang, and Jenq-Neng Hwang. SAMURAI: Motion-Aware Memory for Training-Free Visual Object Tracking With SAM 2. *IEEE Transactions on Image Processing*, 35:970–982, 2026. doi: 10.1109/TIP.2026.3651835.

Tianyu Yang and Antoni B. Chan. Learning Dynamic Memory Networks for Object Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Fangyi Zhang, Qiang Wang, and Bo Li. PySOT: SenseTime research platform for single object tracking. https://github.com/STVIR/pysot, 2019a. Accessed September 2026.

Lichao Zhang, Abel Gonzalez-Garcia, Joost van de Weijer, Martin Danelljan, and Fahad Shahbaz Khan. Learning the Model Update for Siamese Trackers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019b.

Luka Čehovin Zajc. A modular toolkit for visual tracking performance evaluation. *SoftwareX*, 12: 100623, 2020. ISSN 2352-7110. doi: https://doi.org/10.1016/j.softx.2020.100623.

A Extended Results

We extend the main model comparison from Tables 1 to 3. Tables 5 to 7 add three further benchmarks (GOT-10k, DAVIS_{2017} and YouTube-VOS), together with the throughput of every method on both VOT and VOS. We also report the compared methods in their original SAM 2 implementations alongside SAM2RL, our policy trained on the SAM 2 backbone. We discuss SAM 2 results in Section A.1, the three additional benchmarks in Section A.2, and the VOS throughput in Section A.3.

A.1 SAM2RL Performance

In addition to the SAM 3 backbone results of Tables 1 to 3, we report the performance of compared methods in their original SAM 2 implementations. The results are presented alongside those of SAM2RL in Tables 5 to 7, where SAM2RL is trained in the same way as SAM3RL. The MDP, the controller architecture, the LaSOT training corpus, the PPO hyperparameters, and the inference stride are unchanged, and only the frozen backbone differs. Unlike Section 4, this comparison includes GOT-10k, DAVIS_{2017} and YouTube-VOS. The SAM 2 backbone leaves room for memory control to make a difference, and the compared methods still separate, which does not hold on SAM 3; see Section A.2 for further discussion.

On the VOT(S) challenges and VOT datasets, SAM2RL improves over SAM 2 on all nine benchmarks and ranks first on five. The gains are largest on the challenges, where it is the best method on four of five and second only to SAMITE on VOT-LT_{2021}. On DiDi it is ahead of every hand-crafted rule, while on LaSOT and $\text{LaSOT}_{\text{ext}}$ it trails SAMITE by less than one point. GOT-10k is the only tracking benchmark where the gain is negligible. SAM2RL also runs faster than SAM 2 itself and than every compared method, for the same reason as on SAM 3; see Section 4. On the segmentation benchmarks, SAM2RL ranks first on all six. It is the only method that improves on DAVIS_{2017}. On YT-VOS_{2019}, it matches the baseline, as does SAM2Long, while every other rule falls below it. Unlike on SAM 3, most hand-crafted rules do improve SAM 2 on SA-V, which supports the observation that they were designed to handle the specific weaknesses of the older backbone.

Table 5: **Extended performance on VOT(S) challenges.** We report standard metrics for evaluated challenges. Deltas indicate improvement or decline relative to SAM 2 and SAM 3. **Bold** = best.

		Quality			F-score	
Model	FPS	VOTS_{2026}	VOTS_{2025}	VOTS-RT_{2025}	VOT-LT_{2021}	VOT-LT_{2022}
SAM 3	15.3	66.4	73.4	69.1	77.1	62.8
SAMURAI3	15.1	64.0 -2.4	72.2 -1.2	71.5 +2.4	77.4 +0.3	63.4 +0.6
SAM3Long	9.3	67.9 +1.5	75.2 +1.8	–	77.1 ±0.0	64.5 +1.7
DAM4SAM3	14.2	66.7 +0.3	73.3 -0.1	69.5 +0.4	77.6 +0.5	63.4 +0.6
SAMITE3	12.7	65.4 -1.0	73.6 +0.2	72.7 +3.6	78.3 +1.2	66.2 +3.4
HiM3SAM	11.8	63.0 -3.4	72.5 -0.9	58.8 -10.3	77.1 ±0.0	62.3 -0.5
SAM3RL (ours)	**16.0**	**70.0** +3.6	**76.9** +3.5	**75.1** +6.0	**79.0** +1.9	**68.3** +5.5
SAM 2.1 L	25.1	57.9	65.6	65.6	66.1	55.1
SAMURAI	21.3	59.9 +2.0	68.2 +2.6	68.1 +2.5	75.8 +9.7	61.2 +6.1
SAM2Long	11.0	61.0 +3.1	68.1 +2.5	–	72.9 +6.8	57.5 +2.4
DAM4SAM	16.7	61.7 +3.8	70.4 +4.8	70.5 +4.9	74.5 +8.4	60.9 +5.8
SAMITE	16.3	61.1 +3.2	69.2 +3.6	67.1 +1.5	**78.5** +12.4	63.7 +8.6
HiM2SAM	16.1	58.6 +0.7	67.8 +2.2	60.5 -4.7	73.9 +7.8	62.9 +7.8
SAM2RL (ours)	**25.5**	**63.3** +5.4	**71.4** +5.8	**71.3** +5.7	76.6 +10.5	**64.6** +9.5

A.2 Additional Benchmarks

We additionally report three benchmarks that are not part of the main comparison, GOT-10k, DAVIS_{2017} and YouTube-VOS, in Tables 6 and 7. Performance is scored by the average overlap (AO) on GOT-10k, and by the overall score $\mathcal{G}$ on YouTube-VOS, which is the mean of $\mathcal{J}\&\mathcal{F}$ over categories seen and unseen during training.

Table 6: **Extended performance on VOT datasets.** We report standard metrics for the evaluated datasets. Deltas indicate improvement or decline relative to SAM 2 and SAM 3. **Bold** = best.

	Quality	AUC		AO
Model	DiDi	LaSOT	$\text{LaSOT}_{\text{ext}}$	GOT-10k
SAM 3	71.4	75.9	61.7	84.7
SAMURAI3	71.8 +0.4	76.3 +0.4	63.5 +1.8	**85.0** +0.3
SAM3Long	71.7 +0.3	76.1 +0.2	62.0 +0.3	84.9 +0.2
DAM4SAM3	71.8 +0.4	77.3 +1.4	**65.0** +3.3	84.8 +0.1
SAMITE3	71.8 +0.4	77.6 +1.7	64.7 +3.0	84.5 -0.2
HiM3SAM	70.8 -0.6	75.6 -0.3	61.9 +0.2	84.3 -0.4
SAM3RL (ours)	**72.0** +0.6	**78.4** +2.5	64.7 +3.0	84.8 +0.1
SAM 2.1 L	64.8	69.9	56.9	80.6
SAMURAI	67.8 +3.0	74.8 +4.9	60.0 +3.1	**81.9** +1.3
SAM2Long	66.2 +1.4	73.4 +3.5	59.1 +2.2	81.0 +0.4
DAM4SAM	68.5 +3.7	75.5 +5.6	61.7 +4.8	81.3 +0.7
SAMITE	68.6 +3.8	**75.8** +5.9	**61.9** +5.0	80.1 -0.5
HiM2SAM	67.3 +2.5	74.3 +4.4	60.5 +3.6	75.8 -4.8
SAM2RL (ours)	**69.1** +4.3	75.4 +5.5	61.1 +4.2	80.7 +0.1

Table 7: **Extended performance on VOS datasets.** We report standard metrics for the evaluated datasets. Deltas indicate improvement or decline relative to SAM 2 and SAM 3. **Bold** = best.

		$\mathcal{J}\&\dot{\mathcal{F}}$	$\mathcal{J}\&\mathcal{F}$				$\mathcal{G}$
Model	FPS	MOSE_{v2}	SA-V_{val}	$\text{SA-V}_{\text{test}}$	LVOS_{v2}	DAVIS_{2017}	YT-VOS_{2019}
SAM 3	**15.7**	59.8	82.9	84.6	88.7	**91.6**	**89.6**
SAMURAI3	15.2	56.6 -3.2	78.7 -4.2	81.4 -3.2	88.4 -0.3	91.3 -0.3	89.2 -0.4
SAM3Long	9.4	59.3 -0.5	82.5 -0.4	**85.0** +0.4	89.0 +0.3	91.3 -0.3	89.5 -0.1
DAM4SAM3	14.5	58.0 -1.8	80.4 -2.5	83.1 -1.5	88.9 +0.2	**91.6** ±0.0	89.4 -0.2
SAMITE3	12.8	53.4 -6.4	81.1 -1.8	82.2 -2.4	89.3 +0.6	90.7 -0.9	88.9 -0.7
HiM3SAM	10.6	50.0 -9.8	81.1 -1.8	80.9 -3.7	86.6 -2.1	89.9 -1.7	88.5 -1.1
SAM3RL (ours)	15.4	**60.9** +1.1	**85.4** +2.5	84.6 ±0.0	**89.5** +0.8	91.5 -0.1	89.5 -0.1
SAM 2.1 L	**25.0**	49.4	78.4	79.8	84.2	89.8	**88.5**
SAMURAI	21.1	48.1 -1.3	79.3 +0.9	79.5 -0.3	86.3 +2.1	89.6 -0.2	88.1 -0.4
SAM2Long	12.1	50.7 +1.3	81.4 +3.0	80.7 +0.9	85.0 +0.8	89.8 ±0.0	**88.5** ±0.0
DAM4SAM	17.9	50.5 +1.1	81.1 +2.7	81.9 +2.1	86.8 +2.6	88.4 -1.4	88.3 -0.2
SAMITE	17.0	47.0 -2.4	79.9 +1.5	80.1 +0.3	86.5 +2.3	88.2 -1.6	87.9 -0.6
HiM2SAM	13.9	42.7 -6.7	80.6 +2.2	80.2 +0.4	82.3 -1.9	87.1 -2.7	87.4 -1.1
SAM2RL (ours)	24.1	**52.4** +3.0	**82.5** +4.1	**82.8** +3.0	**86.9** +2.7	**90.7** +0.9	**88.5** ±0.0

GOT-10k (Huang et al., 2021) is the shortest of all tracking benchmarks we evaluate on, with an average sequence length of 127 frames. In Table 6, five of the seven SAM 3 methods lie within 0.3 of each other, and the SAM 3 baseline is already at 84.7. SAM3RL reaches 84.8, 0.2 behind SAMURAI3. This further highlights the previous findings that GOT-10k difficulty is shrinking for modern trackers (Videnović et al., 2026).

DAVIS_{2017} (Pont-Tuset et al., 2018; Perazzi et al., 2016) and **YouTube-VOS** (Xu et al., 2018) are the classic VOS benchmarks, on which modern methods are near 90 $\mathcal{J}\&\mathcal{F}$ and which contain mostly salient, dominant and isolated objects (Ding et al., 2025a). In Table 7, the SAM 3 baseline already performs well on these datasets, leaving little room for memory-control methods to improve it. Therefore, all methods remain within a few points of each other, with the main exception of HiM3SAM. The small differences on these benchmarks contrast with the larger gaps on other VOS

datasets, indicating that conventional short VOS benchmarks do not provide a discriminative test of memory management.

Note that the benchmark counts in the main paper, such as "first on 10 of 12", are over the twelve benchmarks of Tables 1 to 3 and exclude the three discussed here, because on SAM 3, none of the three benchmarks separates the compared methods. We nonetheless keep them in Tables 6 and 7, since on the weaker SAM 2 backbone they still separate the methods, and Section A.1 evaluates SAM2RL performance over all fifteen.

A.3 VOS THROUGHPUT

In addition to the VOTS throughput reported in Section 4, we measure VOS throughput on MOSE_{v2} and list it in Table 7. The protocol is the same as on VOTS, but MOSE_{v2} videos are annotated at an average of 5 FPS, so both SAM2RL and SAM3RL update their memory on every frame with $s = 1$, unlike every-6-frames updates on VOTS with $s = 6$. SAM3RL falls slightly behind SAM 3 at 15.4 vs 15.7 FPS, but remains faster than all other methods. The same holds for SAM2RL, which runs at 24.1 against 25.0 FPS for SAM 2.1 L, and is the fastest after the baseline.

B Tracker Evaluation Details and Reproducibility

Benchmarks and metrics. Table 8 lists every dataset we evaluate on, with the number of sequences, objects, frame rate, and length. Performance is measured by the tracking quality (Kristan et al., 2023b) on the VOTS challenges and on DiDi, and by the F-score (Lukežič et al., 2021) on the VOT-LT challenges. On LaSOT and $\text{LaSOT}_{\text{ext}}$, it is measured by the area under the success rate curve (AUC) (Wu et al., 2015). On VOS benchmarks, it is measured by $\mathcal{J}\&\mathcal{F}$ (Pont-Tuset et al., 2018), the mean of the region similarity $\mathcal{J}$ and the contour accuracy $\mathcal{F}$. MOSE_{v2} replaces $\mathcal{F}$ by an adaptive boundary accuracy $\dot{\mathcal{F}}$ that rescales the tolerance with object size.

We score LaSOT and $\text{LaSOT}_{\text{ext}}$ with the PyTracking evaluation code (Danelljan et al., 2019) (commit `9cba77e`), which implements the AUC protocol of the official LaSOT toolkit (Fan et al., 2019). Our SAM 2.1 runs reproduce the published AUC of 70.0 on LaSOT and 56.9 on $\text{LaSOT}_{\text{ext}}$ (Ding et al., 2025b; Videnović et al., 2026) to within 0.1. Some evaluators implement the protocol differently (e.g., the got10k toolkit (Huang et al., 2021) and the pysot toolkit (Zhang et al., 2019a)), as does the evaluation code in the SAMURAI codebase (Yang et al., 2026). For instance, on $\text{LaSOT}_{\text{ext}}$, their implementations inflate scores of every method we report by up to 1.7 AUC compared to the official protocol. GOT-10k, MOSE_{v2} and all VOTS challenges are scored on their evaluation servers. DiDi and VOT-LT are scored with the VOT toolkit (Čehovin Zajc, 2020).

Table 8: **Evaluation benchmarks** with VOT(S) challenges (top), VOT datasets (middle), VOS datasets (bottom). Seq. and Obj. are the numbers of sequences and annotated targets, respectively. FPS is the annotation frame rate, Frames is the average number of frames per sequence, and s is the stride defined in Section 3.2. The three length columns give the average, shortest and longest sequence duration in seconds. SA-V videos are recorded at 24 FPS, but annotated at 6.

						Length (s)		
Dataset	Seq.	Obj.	FPS	s	Frames	Avg.	Min	Max
VOTS_{2026} (Kristan et al., 2026)	170	375	30	6	1860	62.0	2.1	356.6
VOTS(-RT)_{2025} (Kristan et al., 2023b; 2025)	144	341	30	6	2074	69.1	2.1	356.6
VOT-LT_{2021} (Kristan et al., 2021)	50	50	30	6	4306	143.5	46.3	990.0
VOT-LT_{2022} (Kristan et al., 2023a)	50	50	30	6	3366	112.2	32.6	356.6
DiDi (Videnović et al., 2026)	180	180	30	6	1527	50.9	2.1	333.3
LaSOT (Fan et al., 2019)	280	280	30	6	2448	81.6	33.3	333.3
$\text{LaSOT}_{\text{ext}}$ (Fan et al., 2021)	150	150	30	6	2394	79.8	66.8	90.1
GOT-10k (Huang et al., 2021)	180	180	10	2	127	12.7	5.1	92.0
MOSE_{v2} (Ding et al., 2025a)	433	575	5	1	154	30.7	4.8	488.0
SA-V_{val} (Ravi et al., 2025)	155	293	24 / 6	5	414	17.2	2.2	38.4
$\text{SA-V}_{\text{test}}$ (Ravi et al., 2025)	150	278	24 / 6	5	426	17.7	2.7	36.1
LVOS_{v2} (Hong et al., 2024)	140	239	6	1	472	78.6	19.2	356.7
DAVIS_{2017} (Pont-Tuset et al., 2018)	30	61	24	5	67	2.8	1.4	4.3
$\text{YouTube-VOS}_{2019}$ (Xu et al., 2018)	507	1063	6	1	27	4.5	0.7	6.0

Previously published results for the compared methods and baselines are listed in Table 9 and Table 10, next to our own measurements. Most agree within noise, while some differ more. We attribute these differences to evaluation protocols that vary between papers (e.g., mentioned $\text{LaSOT}_{\text{ext}}$ case), to unspecified prompt masks, and to differences in hardware and libraries. Interestingly, even SAM 2 baseline numbers are not consistent between previous work.

Initialization prompts used for the trackers in our evaluation are as follows. We use segmentation masks wherever such are available. A mask localizes the target more precisely than a box. SAM 2 reports 79.3 against 74.4 $\mathcal{J}\&\mathcal{F}$ averaged over 17 video datasets when the first frame is prompted by a ground-truth mask rather than a box (Ravi et al., 2025). SAM 3 then inherits masks as main prompts in its evaluation (Carion et al., 2026).

We do not use mask prompting only where a method or a challenge protocol prescribes otherwise. On VOS benchmarks, every method is initialized with the ground-truth mask of the dataset. On

Table 9: **Performance on tracking benchmarks across previous work.** For each work, we list the numbers for baselines as reported there (whether independently measured, or cited from prior work). The last block reports our own measurements. Hyphen means the benchmark was not reported.

		Quality		AUC		AO
Source	Model	DiDi	VOTS_{2025}	LaSOT	$\text{LaSOT}_{\text{ext}}$	GOT-10k
SAMURAI (Yang et al., 2026)	SAM 2.1 L	–	–	68.5	58.6	–
	SAMURAI	–	–	74.2	61.0	81.7
SAM2Long (Ding et al., 2025b)	SAM 2.1 L	–	–	70.0	56.9	80.7
	SAM2Long	–	–	73.9	60.9	81.1
SAMITE (Xu et al., 2025)	SAM 2.1 L	–	–	68.5	58.6	–
	SAMURAI	–	–	74.2	61.0	–
	SAMITE	–	–	74.7	62.1	–
HiM2SAM (Chen et al., 2026)	SAM 2.1 L	–	–	68.5	58.6	–
	SAMURAI	–	–	74.2	61.0	–
	DAM4SAM	–	–	75.1	60.9	–
	SAM2Long	–	–	73.9	60.9	–
	HiM2SAM	–	–	75.1	62.8	–
DAM4SAM (Videnović et al., 2026)	SAM 2.1 L	64.9	66.1	70.0	56.9	80.7
	SAMURAI	68.0	–	–	–	–
	DAM4SAM	69.4	71.1	75.1	60.9	81.1
	SAM2Long	64.6	–	–	–	–
Ours	SAM 2.1 L	64.8	65.6	69.9	56.9	80.6
	SAMURAI	67.8	68.2	74.8	60.0	81.9
	DAM4SAM	68.5	70.4	75.5	61.7	81.3
	SAM2Long	66.2	68.1	73.4	59.1	81.0
	SAMITE	68.6	69.2	75.8	61.9	80.1
	HiM2SAM	67.3	67.8	74.3	60.5	75.8
	SAM 3	71.4	73.4	75.9	61.7	84.7

LaSOT, $\text{LaSOT}_{\text{ext}}$ and GOT-10k, we use the masks from the set released by Videnović et al. (2026), so that the prompt is reproducible from a public file. The single exception for these VOT datasets is SAMURAI(3), which we prompt with the box, as its authors do and as its published numbers were obtained.[1] DiDi ships its own first-frame masks, and we use them. On VOT-LT_{2021} and VOT-LT_{2022}, we prompt with the ground-truth box. That is the official protocol of both challenges, and no public mask set exists either. On VOTS the toolkit delivers the ground-truth mask of each target.

Object handling. We evaluate trackers in a per-object rather than joint setup. Each object is initialized with its first non-empty annotation and propagated forward from that frame. The tracker state is reset between objects. This is the only mode all methods support, since SAMURAI, DAM4SAM, SAM2Long, SAMITE and HiM2SAM are single-object trackers by construction. It is also the better mode for SAM 3, which gains up to 2.8 $\mathcal{J}\&\mathcal{F}$ over joint tracking on $\text{SA-V}_{\text{test}}$. VOTS multi-target sequences are run in the sequential single-object mode of the toolkit, one tracker process per target.

Ports implementation. None of the compared methods has an official SAM 3 implementation, so we port each to that backbone. We replace the SAM 2 video predictor with the SAM 3 tracker and leave the method's code, hyperparameters, and memory layout unchanged. Additionally, we implement the striding described in Section 3.2 for DAM4SAM (Videnović et al., 2026). This follows its VOS protocol, which Videnović et al. (2026) describe in the paper. The official code for this feature was never released, so we reproduce it from the description. We attach the port implementations in the supplementary material to the submission.

[1] We also ran SAMURAI with the mask prompt on LaSOT. The box arm is better by 0.3 AUC.

Table 10: **Performance on VOS datasets across previous work.** For each work, we list the numbers for baselines as reported there (whether independently measured, or cited from prior work). The last block presents our own measurements. Hyphen means the benchmark was not reported.

		$\mathcal{J}\&\dot{\mathcal{F}}$	$\mathcal{J}\&\mathcal{F}$				$\mathcal{G}$
Source	Model	MOSE_{v2}	SA-V_{val}	$\text{SA-V}_{\text{test}}$	LVOS_{v2}	DAVIS_{2017}	YT-VOS_{2019}
MOSE_{v2} (Ding et al., 2025a)	SAM 2.1 L	49.4	–	78.4	84.0	90.7	89.3
	SAMURAI	51.1	–	–	–	–	–
	DAM4SAM	51.2	–	–	–	–	–
	SAM2Long	51.5	–	81.2	85.3	88.8	90.2
DAM4SAM (Videnović et al., 2026)	SAM 2.1 L	–	–	–	80.6	–	–
	DAM4SAM	–	–	–	85.5	–	–
SAM2Long (Ding et al., 2025b)	SAM 2.1 L	–	78.6	79.6	84.0	–	–
	SAM2Long	–	81.1	81.2	85.3	–	–
SAM3 (Carion et al., 2026)	SAM 2.1 L	47.9	77.9	78.4	79.6	90.7	–
	SAMURAI	51.1	79.8	80.0	84.2	89.9	–
	SAM2Long	51.5	81.2	81.1	85.9	91.4	–
	SAM3	60.3	83.5	84.4	88.5	92.2	–
Ours	SAM 2.1 L	49.4	78.4	79.8	84.2	89.8	88.5
	SAMURAI	48.1	79.3	79.5	86.3	89.6	88.1
	DAM4SAM	50.5	81.1	81.9	86.8	88.4	88.3
	SAM2Long	50.7	81.4	80.7	85.0	89.8	88.5
	SAM3	59.8	82.9	84.6	88.7	91.6	89.6

SAM2Long and SAM3Long are not online trackers. Ding et al. (2025b) keeps several memory pathways and decodes the video only after tracking ends, along the pathway that leads in the last frame. We preserve this behavior for both SAM2Long and SAM3Long, since changing them to fulfil the causality would be a different algorithm. However, the comparison on the tracking benchmarks is uneven, where every other method, ours included, is online. As a result, SAM(2/3)Long are excluded from VOTS-RT_{2025}, where the answer for frame t is due before frame $t + 1$ arrives.

Hardware. All evaluations are computed on NVIDIA A100 40 GB GPUs unless specified otherwise. Throughput is also measured on the A100, as the mean over three runs. It times only model inference, with all the data preprocessing and postprocessing being excluded.

Notably, some methods utilize much more RAM or GPU memory than others. For instance, the longest VOT-LT_{2021} sequence, decoded to CPU, takes 348 GB for all methods. However, both SAM(2/3)Long need 600 GB on that sequence, since they store three pathways per frame. Similarly, the SAMITE prototype bank of up to 30 frames exceeds the 40 GB of the A100 on MOSE_{v2}, so the SAMITE family is run on NVIDIA H200 NVL. Throughput is nonetheless measured on the A100 for SAMITE and SAM2Long, since the sequences used for timing fit into its memory.

C RACE CONDITION IN SAM 3

SAM 3 video predictor offloads per-frame tracking state to pageable host memory with `non_blocking=True` and then reads that buffer on the CPU without synchronizing, which PyTorch documents as undefined behavior. Within a single `add_new_points_or_box` call, the asynchronous copy of the conditioning-frame mask is launched and the same tensor is interpolated on the host a few lines later. If the copy did not land, the host will read the leftovers of the previous sequence in the process. The corrupted mask is encoded into the memory bank as frame 0, and because long-term tracking never re-initializes, the sequence is lost from frame 1 rather than degrading over time. The failure requires box initialization, CPU state offloading, and a preceding sequence in the same process. We fix it by forcing CUDA→CPU copies to be synchronous.

The impact of the race condition was seen most severely on the VOT-LT$_{2021}$ challenge, where all methods are box-prompted, and on SAMURAI3, which is also box-prompted on all benchmarks; see Appendix B. Table 11 shows VOT-LT$_{2021}$ before and after the fix. Four baselines gain between 5.1 and 12.3 F. DAM4SAM3 and SAMITE3 are byte-identical on all fifty sequences before and after the fix. We report performance changes for SAMURAI3 in Table 12. Post-fix, SAMURAI3 recovers $+13.1$ AUC on LaSOT and $+7.7$ on LaSOT$_{\text{ext}}$. All other numbers in this work are post-fix.

Table 11: **Effect of forcing CUDA→CPU copies to be synchronous on VOT-LT$_{2021}$.** DAM4SAM3 and SAMITE3 were unaffected.

Method	pre-fix	after	Δ
HiM3SAM	64.8	77.1	+12.3
SAM3Long	69.1	77.1	+8.0
SAM 3	71.4	77.1	+5.7
SAMURAI3	72.3	77.4	+5.1
DAM4SAM3	77.6	77.6	±0.0
SAMITE3	78.3	78.3	±0.0

Table 12: **SAMURAI3 performance on the VOT datasets exposed to the race condition before and after the fix.** Unlike other trackers, SAMURAI3 is always prompted by bounding boxes; see Appendix B. Therefore, it was the only one affected outside VOT-LT$_{2021}$.

Split	Metric	pre-fix	after	Δ
GOT-10k	AO	82.8	85.0	+2.2
LaSOT	AUC	63.2	76.3	+13.1
LaSOT$_{ext}$	AUC	55.8	63.5	+7.7
VOT-LT$_{2021}$	F	72.3	77.4	+5.1

D TRAINING SETUP

We provide further details on the training setup in addition to those in Section 3. All training hyperparameters are listed in Table 13.

Pre-computed embeddings. The image encoder is frozen and its output does not depend on the policy. Thus, we compute the image embeddings of the whole training corpus once and cache them on disk. During rollouts, only the tracker components after the image encoder are run.

Batch composition. Each rank runs 64 parallel environments, each tracking one object. They come from 16 videos, with each object copied four times. We truncate the batch to the length of its shortest video, so that all environments are synchronized and process the same frame index at every step. To discard few frames, videos are grouped into batches by similar length, with each length jittered by up to 16 steps so that the same videos are not always grouped together.

Truncation. Apart from this batch truncation, episodes are not capped, whereas a rollout collects 376 steps. An episode that crosses the rollout boundary continues in the next iteration, and at the cut its return is bootstrapped with the critic's value of the last state. The end of the truncated batch, by contrast, is terminal and is not bootstrapped.

Table 13: **Training hyperparameters for SAM2RL and SAM3RL.** Both backbones are frozen. Only the actor–critic is optimized on pre-extracted image embeddings.

Hyperparameter	Value
Task	
Backbones	SAM 3, frozen; image size 1008×1008, bf16
	SAM 2.1 L, frozen; image size 1024×1024, bf16
Memory bank size	7 slots
Action space	7 (evict one slot after appending the current frame)
Trainable parameters	6.45 M (actor 3.23 M, critic 3.22 M)
Data	
Corpus	$\text{LaSOT}_{\text{5fps}}$ (stride-6 subsample of LaSOT, 5 fps)
Split	896 train / 224 validation sequences (of 1120)
Episode length	uncapped (full video; no length transform)
Batch composition	similar-length batching, length jitter 16
Environments per rank	64, drawn from 16 unique videos (object multiplier 4)
PPO	
Iterations	400
Parallel environments	256 (64×4 ranks)
Rollout length T	376 steps (96,256 transitions per iteration)
Update epochs	4
Minibatch size	1024
Optimizer	Adam, $\varepsilon = 10^{-5}$, weight decay 0
Learning rate	3×10^{-4}, constant (no annealing)
Discount γ	1.0
Task horizon	finite (episode end is terminal)
Advantage estimator	GAE, $\lambda = 1$ (Monte-Carlo return minus critic baseline)
Policy clip ϵ	0.1
Target KL (early stop)	0.02
Entropy coefficient	10^{-4}, constant (no annealing)
Value loss	Huber, $\delta = 5$; coefficient 0.5
Advantage normalization	yes (per minibatch)
Gradient clipping	$\|g\|_2 \leq 2.0$, applied per head
Compute	
Hardware	24 hours on $4\times$ H200 (one node, 32 CPU, 256 GB), DDP
Seeds	1, 2, 3 (main results use seed 1, the best on local validation)

E ABLATIONS

All ablations are based on the SAM3RL training setup described in Section 3 and Appendix D, and the evaluation protocols follow Section 4. We report ablation results as the mean over three training seeds to reduce the impact of training variance. In contrast, the main results in Tables 1 to 3 use the single best of these seeds, selected on our validation split without using the test benchmarks, as they aim to show the best model we can obtain. The effects of the actor input features, the object pointer selection rule, and the reward normalization are presented in Section E.1, Section E.2, and Section E.3, respectively.

E.1 ACTOR INPUT FEATURES

The state s_t groups the per-slot features into four disjoint groups: temporal, scores, semantics, and geometry; see Section 3.1. We test whether each group is necessary or sufficient with delete-one and keep-one ablations, respectively. The former removes one group from the actor input, while the latter keeps exactly one. Everything else, including the critic input, is unchanged. The groups differ in the input size, with the baseline actor input having 1,560 dimensions after the pointer projection, of which semantics take 1,152, temporal 256, geometry 120, and scores 32. The performance is presented in Tables 14 and 15 as a mean over three seeds for each arm. No single group is critically necessary, and no single group is sufficient either. All arms still improve over SAM 3, except the keep-one arms on MOSE_{v2}, yet none matches the all-inputs actor across all benchmarks.

Table 14: **Delete-one actor input feature group ablation.** Each row removes one feature group from the actor input. Everything else, including the critic input, is unchanged. Every entry is the mean over three seeds, with the standard error of the mean in superscript. Cell colour encodes the difference to the full-input SAM3RL mean, green for a gain and red for a drop, darker for a larger difference. **Bold** indicates the best.

	Quality	F-score	AUC		$\mathcal{J}\&\mathcal{F}$
Actor input	VOTS_{2026}	VOT-LT_{2022}	LaSOT	$\text{LaSOT}_{\text{ext}}$	MOSE_{v2}
SAM3RL (all inputs)	**69.5**$_{\pm 0.2}$	68.5$_{\pm 0.3}$	**78.0**$_{\pm 0.3}$	**64.6**$_{\pm 0.1}$	60.6$_{\pm 0.3}$
w/o temporal	69.1$_{\pm 0.3}$	**68.9**$_{\pm 0.5}$	**78.0**$_{\pm 0.1}$	64.3$_{\pm 0.3}$	**60.7**$_{\pm 0.3}$
w/o scores	69.2$_{\pm 0.2}$	68.4$_{\pm 0.4}$	77.9$_{\pm 0.1}$	64.1$_{\pm 0.1}$	60.4$_{\pm 0.1}$
w/o semantics	69.1$_{\pm 0.4}$	67.3$_{\pm 0.4}$	**78.0**$_{\pm 0.2}$	64.5$_{\pm 0.2}$	60.2$_{\pm 0.2}$
w/o geometry	69.2$_{\pm 0.2}$	68.2$_{\pm 0.2}$	77.8$_{\pm 0.1}$	64.5$_{\pm 0.2}$	59.7$_{\pm 0.2}$

Table 15: **Keep-one actor input feature group ablation.** Each row keeps one feature group in the actor input and removes the other three. Everything else, including the critic input, is unchanged. Every entry is the mean over three seeds, with the standard error of the mean in subscript. Cell colour encodes the difference to the full-input SAM3RL mean, green for a gain and red for a drop, darker for a larger difference. **Bold** indicates the best.

	Quality	F-score	AUC		$\mathcal{J}\&\mathcal{F}$
Actor input	VOTS_{2026}	VOT-LT_{2022}	LaSOT	$\text{LaSOT}_{\text{ext}}$	MOSE_{v2}
SAM3RL (all inputs)	**69.5**$_{\pm 0.2}$	68.5$_{\pm 0.3}$	78.0$_{\pm 0.3}$	64.6$_{\pm 0.1}$	**60.6**$_{\pm 0.3}$
only temporal	68.7$_{\pm 0.3}$	68.3$_{\pm 0.0}$	77.6$_{\pm 0.2}$	64.1$_{\pm 0.1}$	59.2$_{\pm 0.1}$
only scores	68.7$_{\pm 0.2}$	66.3$_{\pm 0.2}$	77.7$_{\pm 0.2}$	**64.8**$_{\pm 0.1}$	60.1$_{\pm 0.6}$
only semantics	69.1$_{\pm 0.4}$	**68.8**$_{\pm 0.3}$	**78.1**$_{\pm 0.3}$	63.7$_{\pm 0.4}$	59.8$_{\pm 0.2}$
only geometry	68.8$_{\pm 0.2}$	67.6$_{\pm 0.6}$	77.7$_{\pm 0.2}$	63.9$_{\pm 0.2}$	58.9$_{\pm 0.8}$

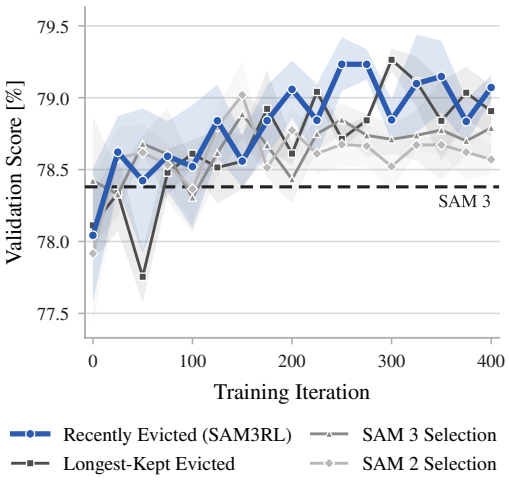

Figure 6: **Object pointer selection.** Validation score on the LaSOT validation split during SAM3RL training; mean over three seeds with a ±SEM band. The arms differ only in the rule that fills the nine object pointers outside the bank.

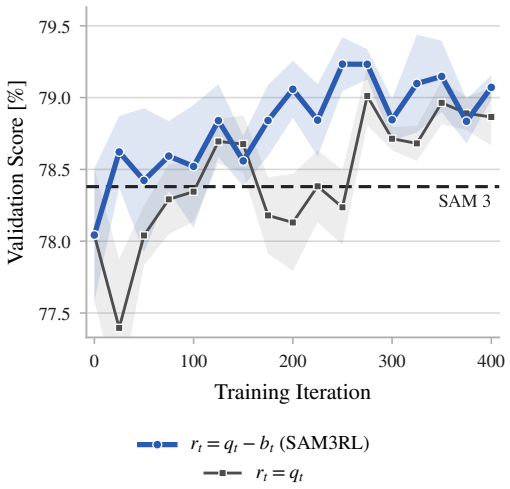

Figure 7: **Reward normalization.** Validation score on the LaSOT validation split during training; mean over three seeds with a ±SEM band. The arms differ only in the b_t normalized reward, $r_t = q_t - b_t$ against the raw quality $r_t = q_t$.

E.2 Object Pointer Selection

Besides the seven memory features, the memory attention of SAM 2 and SAM 3 also reads up to sixteen object pointers, one of which always comes from the conditioning frame; see Section 3. SAM 2 takes the remaining pointers from the 15 most recent frames, whereas SAM 3 takes them from the most recent frames whose effective IoU score exceeds a threshold, always including the previous frame. SAM3RL instead uses a *recently evicted* rule, which reads the pointers of the conditioning frame and of the six frames currently in the memory bank, and fills the remaining nine slots with the frames most recently evicted from the bank. We also consider a *longest-kept* rule, which ranks evicted frames by the number of decisions they remained in the memory bank and selects the pointers of the top nine. Unlike the rules of SAM 2 and SAM 3, both of our rules depend on the agent's actions. Figure 6 compares all four rules within SAM3RL, reporting the LaSOT validation score of each rule every 25 training iterations; all other components are kept fixed.

After the first 200 iterations, both our action-dependent rules remain above the SAM 2 and SAM 3 rules, showing that the object pointer selection rule affects tracking performance. Between the two, the recently evicted rule scores higher in most evaluations, but the longest-kept rule performs comparably and their seed bands overlap. Designing better rules could improve performance further. Moreover, since learned memory control outperforms hand-crafted rules for memory bank selection, extending the policy to also select object pointers is a natural direction for future work.

E.3 Reward Normalization

We train SAM3RL against the normalized reward $r_t = q_t - b_t$, which is the difference between the tracking quality q_t of SAM 3 under our learned memory control and the tracking quality b_t of SAM 3 under its native memory control. Since much of the variance of q_t comes from the difficulty of the video rather than from the actions taken, subtracting the policy-independent b_t reduces this variance while leaving the optimal policy unchanged. In Figure 7, we ablate this normalization by training on the raw quality $r_t = q_t$ with everything else unchanged, and evaluate both arms every 25 training iterations on the LaSOT validation split. The raw-reward arm falls below SAM 3 itself on several evaluations in the first 250 iterations and is less stable than the difference-based arm, which stays above SAM 3 throughout.

F EXTENDED ANALYSIS OF LEARNED MEMORY CONTROL

We present the action distributions of SAM 3 and SAM3RL in Figure 8. SAM 3 evicts the oldest frame on almost every decision, while SAM3RL spreads its decisions over all seven actions, with no single action dominating. The admission rate of SAM3RL as a function of object motion and appearance change is shown in Figure 9, where the admission rate rises both with object motion and object appearance change. This finding is consistent with the sequences shown in Figures 5 and 10.

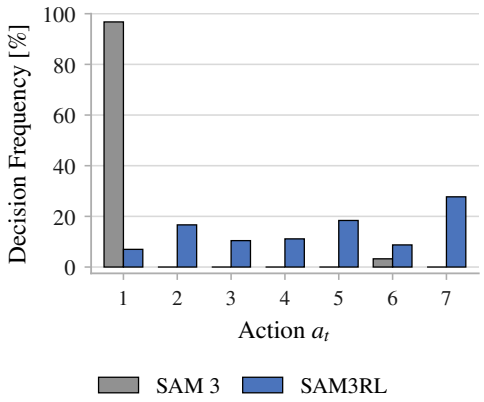

Figure 8: **Action distributions.** Frequency of each action a_t for SAM 3 and SAM3RL on our LaSOT validation split. Action $a_t \in \{1, \ldots, 6\}$ evicts the stored memory $m_{\tau_{a_t}}$, where $a_t = 7$ rejects the incoming frame. The SAM 3 rule uses only two actions. It evicts the most recent stored frame m_{τ_6} ($a_t = 6$) if its effective IoU score falls below the memory threshold, and the oldest frame m_{τ_1} ($a_t = 1$) otherwise.

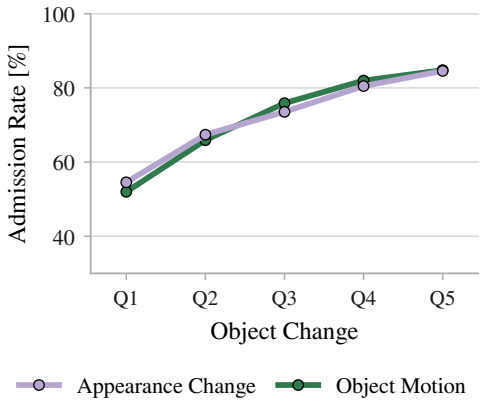

Figure 9: **Admission follows object change.** Share of incoming frames SAM3RL admits per quintile of object motion and appearance change on our LaSOT validation split. *Object motion* is the shift of the ground-truth box center between consecutive decisions, normalized by the square root of the box area. *Appearance change* is the DINOv3 (Siméoni et al., 2025) cosine distance between consecutive ground-truth crops.

Figures 11 to 13 compare the memory banks of SAM 3, DAM4SAM3, and SAM3RL around three absence gaps. In all three, SAM3RL keeps old, temporally sparse frames and admits few new ones while the target is absent, so the pre-absence appearance is still in its bank when the target returns. In Figure 11, this lets SAM3RL reacquire the formula car that both SAM 3 and DAM4SAM3 lose. In Figure 12, SAM3RL reacquires the soldier's hat, which DAM4SAM3 swaps for a distractor. On the other hand, in Figure 13, its bank retains frames showing the longboarder small and centered, but the rider returns large and on the left, so SAM3RL reacquires later than SAM 3 and DAM4SAM3.

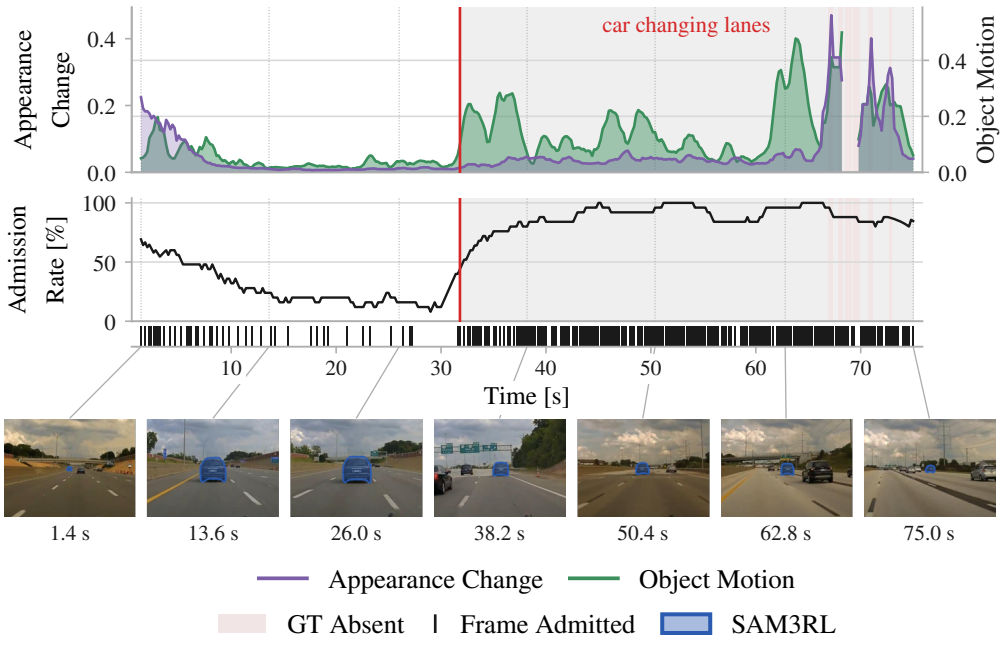

Figure 10: **Admission rate of incoming frames by SAM3RL follows object change II.** A 75 s DiDi video sequence in which a car cruises steadily in one lane and then starts moving quickly, changing lanes until the end. *Object motion* is the shift of the ground-truth box center between consecutive decisions, normalized by the square root of the box area. *Appearance change* is the DINOv3 (Siméoni et al., 2025) cosine distance between consecutive ground-truth crops. The middle panel is the admission rate in a 5 s window, one tick per admitted decision. Admissions are sparse during the cruise and dense in the lane-changing interval (red) when object motion rises.

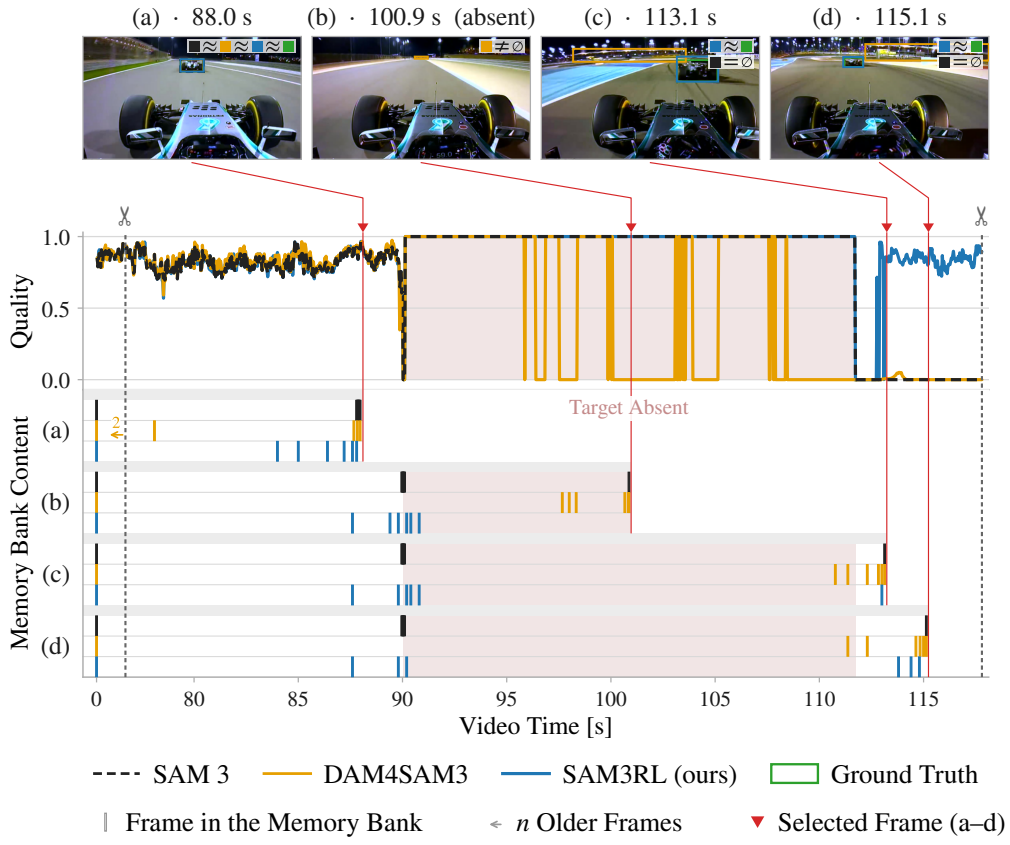

Figure 11: **SAM3RL holds diverse pre-absence frames through a long absence.** A VOT-LT$_{2021}$ sequence captured from the onboard camera of a formula car, in which the car ahead leaves the field of view for about 22 s. The top row shows four selected frames (a–d) with the ground truth and the prediction of each method. The middle panel shows the per-frame tracking quality of each method in the selected time window, with the absence shaded red. The bottom panel shows the memory bank content of each method at the four selected frames, one row per selected frame and method and one tick per stored frame. The prompt frame at 0 s is on the left, and an arrow gives the number of stored frames older than the selected window. SAM 3 correctly predicts no target throughout the absence, with its bank fixed on consecutive frames just before it, in which the car is mostly out of view. It does not reacquire the car after its return. DAM4SAM3 wrongly predicts the target at several points during the absence, and by (b) its bank holds only frames from the absence. After the return, its predictions miss the car. SAM3RL also predicts no target during the absence. When the car leaves, its bank holds frames spread over the last few seconds before the absence and a few from the start of the absence, and at (b) and (c) it still holds nearly all of them. SAM3RL reacquires the car about 1 s after its return, and by (d) half of its bank holds frames from after the return, while the pre-absence frames remain.

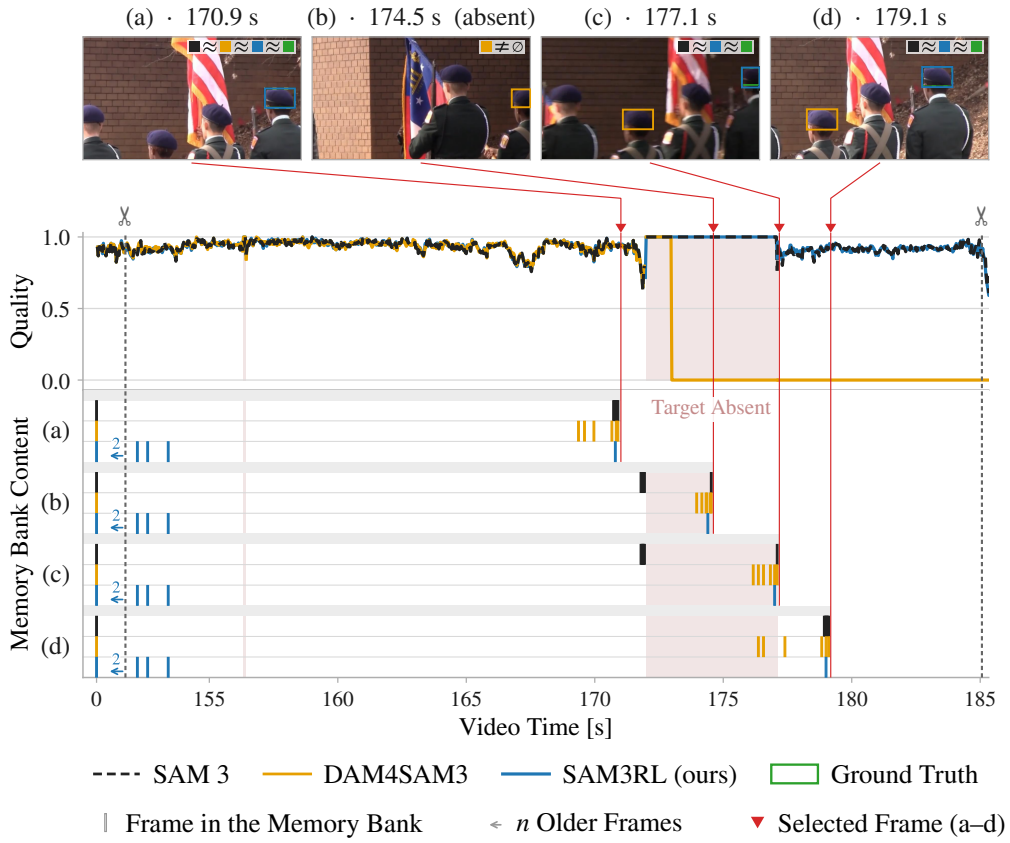

Figure 12: **SAM3RL holds onto very old frames and successfully reacquires.** A LaSOT test sequence of a military ceremony, in which the tracked soldier's hat leaves the field of view for about 5 s among other soldiers wearing near-identical hats. The top row shows four selected frames (a–d) with the ground truth and the prediction of each method. The middle panel shows the per-frame tracking quality of each method in the selected time window, with the absence shaded red. The bottom panel shows the memory bank content of each method at the four selected frames, one row per selected frame and method and one tick per stored frame. The prompt frame at 0 s is on the left, and an arrow gives the number of stored frames older than the selected window. SAM 3 correctly predicts no target throughout the absence, with its bank holding the last five frames before it, and reacquires the hat on the first frame after its return. DAM4SAM3 also predicts no target at the start of the absence, but shortly after it locks onto another soldier's hat. By (b) it has evicted all pre-absence frames, and its bank holds only frames in which it tracks the wrong hat. It keeps following that hat for the rest of the sequence, also after the target returns. SAM3RL correctly predicts no target throughout the absence, but its bank barely changes. When the hat leaves the view, five of its six slots hold frames that are 20–30 s old. These stay in the bank through more than 30 s of tracking, from before (a) until after (d), and only the newest slot turns over. SAM3RL thus keeps the target's appearance in memory throughout the gap and reacquires the hat on the first frame after its return.

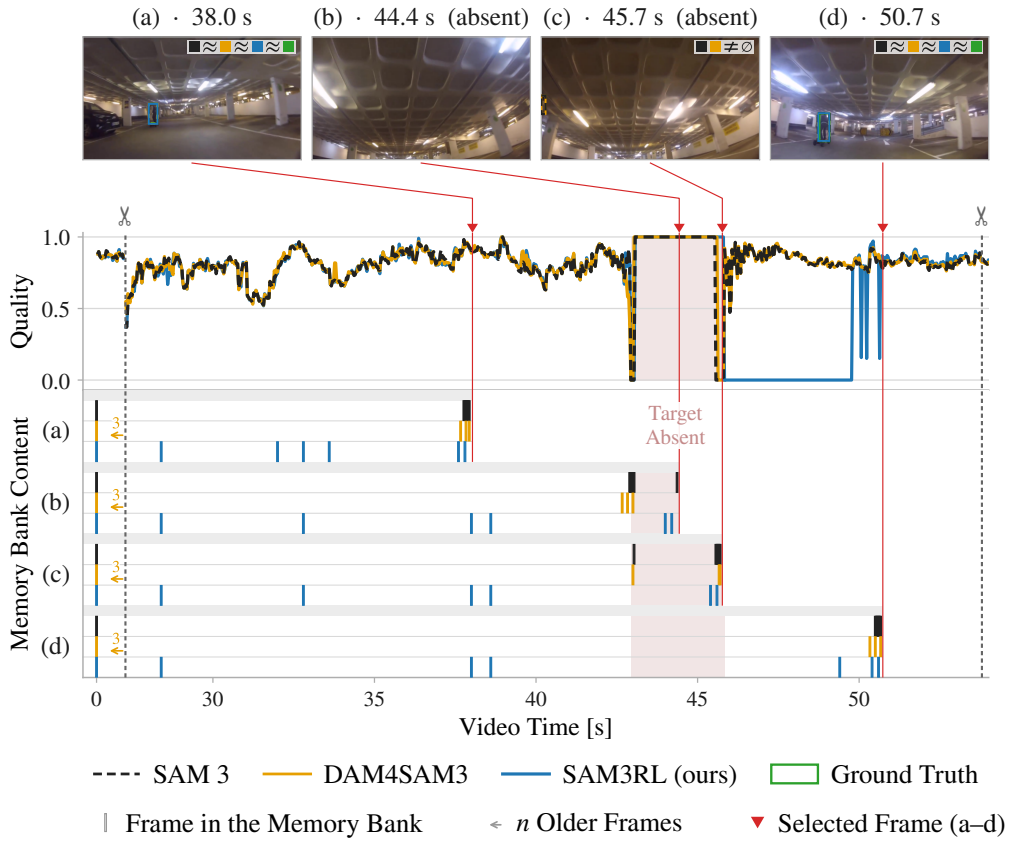

Figure 13: **SAM3RL holds sparse old frames and reacquires late.** A VOT-LT$_{2021}$ sequence capturing a longboarder from behind, who goes out of view on the left for about 3 s. The top row shows four selected frames (a–d) with the ground truth and the prediction of each method. The middle panel shows the per-frame tracking quality of each method in the selected time window, with the absence shaded red. The bottom panel shows the memory bank content of each method at the four selected frames, one row per selected frame and method and one tick per stored frame. The prompt frame at 0 s is on the left, and an arrow gives the number of stored frames older than the selected window. SAM 3 correctly reports the target absent on most frames of the gap and reacquires successfully. DAM4SAM3 keeps 3 old frames and 3 recent ones in all selected frames, although during the absence the 3 recent ones are the pre-absence ones. SAM3RL carries four frames from before the absence through the gap, which are 4–15 s old at the start of the absence and all show the rider small and near the center of the frame, while the rider comes back large and on the left side. The memory bank frames that would match that view all passed through the two newest slots and are not present in the bank of SAM3RL at the end of the absence, but are present in the banks of SAM 3 and DAM4SAM3. SAM3RL fails to reacquire for 4 s after the return, while SAM 3 and DAM4SAM3 reacquire immediately. Once the rider moves back towards the center, SAM3RL reacquires and updates its memory bank.