SEQUENCE-LEVEL POST-TRAINING FOR VISUAL TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual trackers are typically trained with frame-level objectives but deployed with sequential inference, where every prediction determines the next search region. This mismatch limits their ability to optimize the behavior encountered at test time. We present Sequence-level Policy Optimization for Tracking (SPOT), a framework for sequence-level post-training. SPOT represents a tracker's native localization output as a stochastic policy, rolls out a group of trajectories under the same prediction-driven cropping mechanism used at inference, and optimizes trajectory-level IoU returns with a critic-free policy gradient. Multiple trajectories provide relative supervision, while frozen-reference KL regularization constrains the updated tracker near its pretrained behavior. Through a unified policy interface, the same framework applies to one-stream, autoregressive, and memory-based trackers without redesigning or modifying their backbone or prediction-head architectures, or changing their test-time inference pipelines. SPOT consistently improves their supervised base trackers across multiple tracking benchmarks. Code will be released.

1 INTRODUCTION

Visual object tracking is inherently sequential. Given a target in the first frame, a tracker must localize it throughout the remaining video. At test time, each predicted box defines the next search region; an inaccurate prediction can therefore shift future observations and propagate localization errors forward. Tracking performance depends not only on isolated-frame accuracy, but also on whether the target remains locked throughout a prediction-driven sequence. Occlusion, deformation, fast motion, distractors, and out-of-view events make this temporal dependence central to tracking evaluation. (Wu et al., 2013; Smeulders et al., 2014; Marvasti-Zadeh et al., 2022; Javed et al., 2023)

Yet the supervised objectives of many modern trackers remain step-local. Even when trackers use temporal features, dynamic templates, contextual memory, or video-clip autoregressive training as in ARTrackV2, their training search regions are generally constructed from ground-truth target states or anchored at ground-truth boxes, while each step is supervised by a local localization or coordinate-token loss. (Zhang et al., 2019; Mayer et al., 2021; Cui et al., 2022; Cai et al., 2024; Zheng et al., 2024; Bai et al., 2024) Such training can improve temporal representations and current-frame predictions, but it does not directly optimize how early decisions affect subsequent observations, internal states, and long-term tracking performance. At test time, by contrast, search regions and temporal states evolve recursively from the tracker's own predictions. This creates a mismatch in both state distribution and optimization objective, as illustrated in Fig. 1.

Prior work has also explored sequence-level objectives for prediction-driven tracking, highlighting the value of optimizing under state distributions induced by the tracker itself. (Kim et al., 2022) However, what remains underexplored is how to build a unified sequence-level post-training framework that accommodates trackers with heterogeneous localization outputs and temporal states while enabling stable optimization. To address this gap, we introduce Sequence-level Policy Optimization for Tracking (SPOT). SPOT represents each tracker's native localization output as a stochastic policy, rolls out multiple independently evolving trajectories under the same prediction-driven cropping mechanism used at inference, and optimizes trajectory-level IoU returns with a critic-free score-function policy gradient. The trajectories provide relative supervision, while frozen-reference KL regularization constrains the updated tracker to remain close to its pretrained behavior.

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.

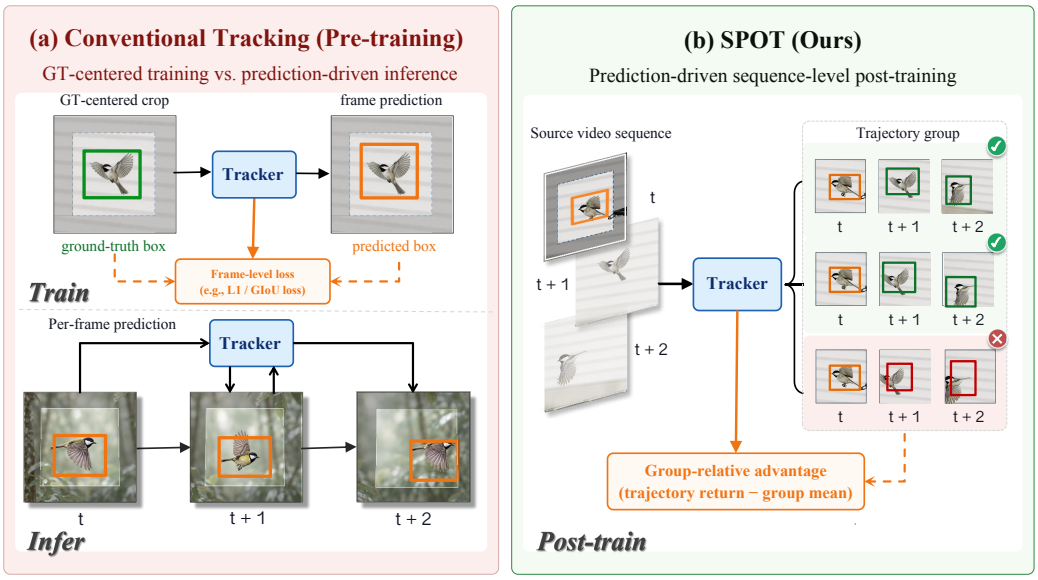

Figure 1: Conventional tracker pre-training and SPOT. (a) Conventional pre-training uses ground-truth-anchored search regions and step-local supervision, without directly optimizing prediction-driven inference. (b) SPOT post-trains the tracker on sampled prediction-driven trajectories using group-relative policy optimization. Gray regions indicate source-frame context outside the active search crop.

We apply SPOT to three representative modern trackers with distinct localization outputs and temporal mechanisms: one-stream OSTrack, autoregressive ARTrackV2, and memory-based MCITrack. (Ye et al., 2022; Bai et al., 2024; Kang et al., 2025) The results show that SPOT serves as a unified post-training stage across these architectures without changing their test-time inference pipelines. Across four tracking benchmarks, SPOT consistently improves the corresponding supervised base trackers, with gains of up to 1.7 AO points on GOT-10k.

We make the following contributions.

- We introduce SPOT, a head-only sequence-level post-training framework that aligns tracker optimization with prediction-driven sequential inference without designing a new tracking architecture.
- SPOT unifies heterogeneous native localization outputs as stochastic policies and optimizes prediction-driven trajectories with a critic-free score-function policy gradient on trajectory-level IoU returns. The score-function estimator handles non-differentiable crop transitions, while group-relative advantages, batch-global normalization, and frozen-reference KL stabilize post-training without changing the original test-time pipelines.
- We validate SPOT on three configurations spanning one-stream, autoregressive, and memory-based trackers. Under matched inference, SPOT improves GOT-10k AO by 0.6–1.7 points, LaSOT AUC by 0.4–0.6 points, LaSOT$_{\text{ext}}$ AUC by 0.6–1.1 points, and TrackingNet AUC by 0.3–0.7 points. Cumulative-prefix analysis further shows that SPOT better preserves localization accuracy over longer tracking horizons, indicating reduced drift accumulation over time.

2 Related Work

Visual tracking methods. Visual tracking has evolved from SiamFC, SiamRPN, and SiamRPN++ template–search matching to online target modeling in MDNet, ATOM, and DiMP, while other Siamese variants introduce adaptive box prediction, distractor-aware matching, and instance detection (Bertinetto et al., 2016; Li et al., 2018; 2019; Nam & Han, 2016; Danelljan et al., 2019; Bhat

et al., 2019; Chen et al., 2020; Zhu et al., 2018; Voigtlaender et al., 2020). More recently, Transformer trackers such as TransT, STARK, and OSTrack strengthen feature interaction and one-stream modeling; SeqTrack and ARTrackV2 formulate localization as autoregressive coordinate generation, while MCITrack explicitly maintains cross-frame context (Chen et al., 2021; Yan et al., 2021a; Ye et al., 2022; Chen et al., 2023; Bai et al., 2024; Kang et al., 2025). Complementary approaches explore probabilistic regression, target-model prediction, bounding-box refinement, robust object modeling, and diffusion-based localization (Danelljan et al., 2020; Mayer et al., 2022; Yan et al., 2021b; Cai et al., 2023; Xie et al., 2024). Despite improvements in localization and temporal modeling, these trackers are still trained mainly with supervised localization or token losses rather than direct optimization of complete trajectories induced by their own predictions.

Reinforcement learning for tracking. Early work formulates tracking as sequential decision making through discrete actions or online policies, and hierarchical strategies further jointly model tracking state and mode selection (Yun et al., 2017; Supancic & Ramanan, 2017; Zhang et al., 2021). Other RL trackers optimize hyperparameters, computation paths, iterative displacements, continuous box actions, or tracking-mode decisions (Dong et al., 2018; Huang et al., 2017; Ren et al., 2018; Chen et al., 2018; Song et al., 2020). SLT rolls a tracker out on its own predictions, uses average overlap as the trajectory return, and adopts a greedy trajectory as a self-critical baseline (Kim et al., 2022; Rennie et al., 2017). Beyond tracking, MIXER and minimum-risk training illustrate optimization of sequence-level task signals over complete sampled outputs (Ranzato et al., 2016; Shen et al., 2016). SPOT extends this sequence-level view by sampling multiple independently evolving trajectories from the same sequence state, using group-relative returns and a frozen reference policy to support heterogeneous modern trackers.

Post-training policy optimization. Policy-gradient methods and REINFORCE estimate updates from sampled actions and returns; natural policy gradients and TRPO constrain updates through policy-space geometry and trust regions, while GAE and PPO improve advantage estimation and stability (Sutton et al., 1999; Williams, 1992; Kakade, 2001; Schulman et al., 2015; 2016; 2017). DPO, GRPO, and REINFORCE++ represent critic-free post-training based on pairwise preferences, grouped relative returns, and global advantage normalization, respectively (Rafailov et al., 2023; Shao et al., 2024; Hu et al., 2025). SPOT combines group-relative advantage construction with batch-global normalization, with the latter following the global-normalization principle of REINFORCE++. It retains tracking-specific prediction-driven rollouts, trajectory IoU returns, the native policy interface, and reference KL; DPO and GRPO are controlled alternative objectives.

3 METHOD

SPOT is a Sequence-Level Post-training framework applied after supervised pretraining. As summarized in Fig. 2, SPOT exposes a tracker's native localization output as a stochastic policy, rolls that policy out under prediction-driven transitions, scores complete trajectories, and updates the localization policy with a constrained optimization engine. The outer framework is shared across trackers; the native action parameterization is architecture-specific. We first describe the policy interface, rollout, and trajectory return, and then present the default critic-free policy-gradient objective.

3.1 TRACKER AS A STOCHASTIC POLICY

We model tracking step t as a state s_t and an action a_t. The state gathers the search region, cropped around the previous prediction, together with the reference information the tracker maintains, namely the initial template and, depending on the architecture, dynamic templates, a template memory, and internal temporal state such as autoregressive features or an SSM hidden state. The action is the complete native localization output used by the tracker. Its decoded box is used for the IoU reward and the next search-region transition. We factorize the action according to the tracker's native output parameterization,

$$\pi_\theta(a \mid s) = \begin{cases} \displaystyle\prod_{e \in \{l,t,r,b\}} p_{\theta,e}(k_e \mid s), & \text{1D boundary policy,} \\ p_\theta(k \mid s)\, p_\theta(u \mid s, k), & \text{2D center policy.} \end{cases} \tag{1}$$

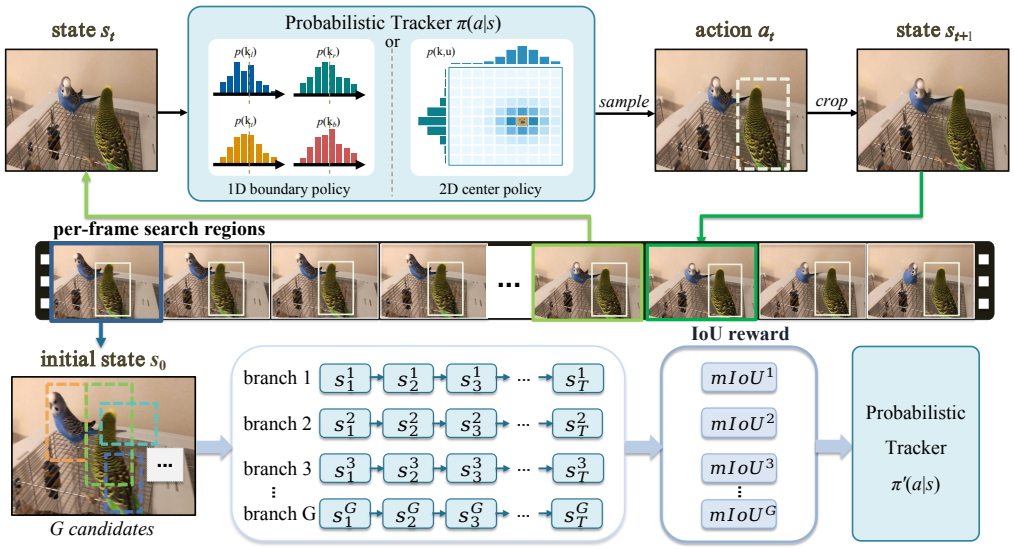

Figure 2: Overview of the SPOT framework. Depending on the architecture, a tracker's native localization output is parameterized either by four one-dimensional boundary distributions or by a two-dimensional center distribution and conditional factors. Sampled actions determine prediction-centered search regions and thereby generate prediction-driven rollouts; G candidate trajectories provide relative trajectory supervision.

For the one-dimensional policy, k_e is the coordinate token selected from the state-conditioned distribution for edge e. For the two-dimensional policy, we write the complete action as $a = (k, u)$, where k is the sampled center cell and u contains the native size and offset factors conditioned on k. In both cases the sampled action is decoded into a box before receiving a reward and driving the next crop.

In addition to sampled actions, each tracker retains its native deterministic readout for monitoring and testing. In our instantiations, OSTrack-b256 and MCITrack-b224 use their native CENTER-style localization heads, which predict a center response together with size and offset outputs. ARTrackV2-b256 follows its original coordinate-token decoding and fusion rule, whereas CENTER trackers select the maximum-response center cell and decode the corresponding native size and offset outputs. The policy adapter therefore requires neither prediction-head replacement nor preliminary alignment training. A frozen copy initialized from the same author-released checkpoint, π_{ref}, serves as the reference policy for KL regularization (Ziegler et al., 2019). Per-architecture details are given in Appendix A.

3.2 Grouped Trajectory Rollout

To optimize test-time behavior we roll out whole trajectories rather than scoring isolated frames. For clarity, we first describe the rollout for one training sequence and introduce the batch index only for global normalization and the final parameter update. The sequence is initialized from an annotated frame. At step t the policy observes the search region cropped around its previous prediction and emits a box. The environment then crops the next search region around that box and updates the dynamic templates, memory, and hidden state following the tracker's own test-time rules. Because the next observation depends on the current action, error accumulation is part of the optimized objective.

This prediction-driven rollout exposes the policy to the state distribution induced by its own behavior rather than keeping it on ground-truth-centered search regions. The motivation is related to data aggregation and train–inference distribution shift in sequential decision making, but no additional expert actions are collected here: after initialization, ground-truth annotations do not determine

state transitions and are used only to evaluate trajectory rewards; every subsequent search region is centered on the branch's own prediction. (Ross et al., 2011; Bengio et al., 2015)

Group rollout. At the initial state s_1, the policy expands the sequence into a group of G branches. Each branch q is rolled out independently for T steps, with its sampled action driving its own crop transition (Fig. 2, bottom). This yields actions $a_{q,t}$ and log-probabilities $\log \pi_\theta(a_{q,t} \mid s_{q,t})$ for $1 \le q \le G$ and $1 \le t \le T$. Because every branch follows its own predictions, the group covers diverse futures of the same sequence; corresponding rollout visualizations are provided in Appendix B.

Trajectory return. The reward is the geometric overlap between the sampled box and the ground-truth annotation, reusing the supervision available in frame-level training; no learned value network is required. We summarize each branch by a temporally weighted trajectory return,

$$R_q^{(\gamma)} = \frac{\sum_{t=1}^{T} \gamma^{t-1} \operatorname{IoU}(a_{q,t},\, g_t)}{\sum_{t=1}^{T} \gamma^{t-1}}, \qquad \gamma > 0, \tag{2}$$

which credits branches that stay well localized over the whole trajectory. The default $\gamma{=}1$ recovers the uniform mean IoU (mIoU); $\gamma{<}1$ discounts later frames, whereas $\gamma{>}1$ places more weight on future frames. We omit the superscript (γ) below for the default $\gamma{=}1$ case.

3.3 Critic-Free Policy-Gradient Optimization

SPOT uses a critic-free score-function policy-gradient update. Unless otherwise stated, the advantage estimator combines group-relative centering with batch-global scale normalization. For each video sequence, the prediction-driven branches form a group and their mean return serves as the local baseline; the centered advantages are then normalized by a single standard deviation pooled over the full distributed training batch. This design combines group-relative advantage construction with the global-normalization principle of REINFORCE++. (Shao et al., 2024; Hu et al., 2025)

Trajectory advantage. During training, let $\mathcal{B}$ denote the set of sequence groups in the distributed optimization batch. For the group under consideration, let $\bar{R} = \frac{1}{G}\sum_{q=1}^{G} R_q$ be its mean return. We normalize $R_q - \bar{R}$ by a single batch-global scale σ_{batch}, computed as the standard deviation of the centered returns pooled over all groups in $\mathcal{B}$. This shared scale avoids normalizing each group by its own variability,

$$A_q = \operatorname{clip}\!\left(\frac{R_q - \bar{R}}{\max(\sigma_{\text{batch}}, \epsilon)},\, -c, c\right). \tag{3}$$

Here, ϵ is a numerical floor for the batch standard deviation, and c is the advantage-clipping threshold.

Reference KL. To keep the policy close to the base tracker, we anchor it to the frozen reference π_{ref} with the component-wise penalty

$$\mathcal{D}_{\text{KL}}(s,a) = \frac{1}{J}\sum_{j=1}^{J} \operatorname{KL}(p_{\theta,j}(\cdot \mid c_j(s,a)) \,\|\, p_{\text{ref},j}(\cdot \mid c_j(s,a))). \tag{4}$$

Here, j indexes the native factors in Eq. (1), $c_j(s,a)$ is the matched context for factor j (the state s for each boundary or center, and (s,k) for center-conditioned size and offset), and $p_{\text{ref},j}$ is the corresponding frozen-reference factor. We use $J{=}4$ for boundaries and $J{=}3$ for center, size, and offset. Discrete and continuous factors use categorical and closed-form Gaussian KL, respectively.

Objective. By the score-function estimator, one trajectory-level advantage multiplies the sum of log-probabilities of all sampled decisions along that trajectory. Broadcasting each trajectory advantage over its T steps, the loss for one sequence group averages the policy-gradient and KL terms over its $G \times T$ sampled branch-steps,

$$\mathcal{L} = -\frac{1}{GT}\sum_{q=1}^{G}\sum_{t=1}^{T} A_q \log \pi_\theta(a_{q,t} \mid s_{q,t}) \;+\; \frac{\beta}{GT}\sum_{q=1}^{G}\sum_{t=1}^{T} \mathcal{D}_{\text{KL}}(s_{q,t}, a_{q,t}). \tag{5}$$

The optimizer further averages this group loss over all sequence groups in the distributed optimization batch. The reward, the advantage, and the reference policy are detached from the gradient, so the only learned object is the policy itself. The update is critic-free and involves no value network. The tracker-specific factorization in Eq. (1) and component-wise KL in Eq. (4) do not change this shared trajectory-level objective.

The update is shared across all trackers. Only the native action factorization, deterministic readout, trainable localization parameters, and carried temporal state are architecture-specific, as detailed in Appendix A. Algorithm 1 summarizes one complete SPOT post-training update.

Algorithm 1 One post-training update of SPOT

1: Sample a distributed batch $\mathcal{B}$ of sequences and construct their initial states
2: Expand each sequence into a group of G rollout branches
3: **for** $t = 1$ to T **do**
4: For every branch, sample $a_t \sim \pi_\theta(\cdot \mid s_t)$
5: Record its log-probability, reference KL, and IoU reward
6: If $t < T$, crop around a_t and update the branch-specific tracker state
7: **end for**
8: Compute trajectory returns and group-relative, batch-normalized advantages
9: Update θ with one AdamW step on Eq. (5)

4 Experiments

4.1 Experimental Setup

Choice of base trackers. We deliberately select three base-tracker families with different localization outputs and temporal states to assess whether the same sequence-level post-training framework applies across these distinct designs. OSTrack-b256 represents a one-stream tracker with center-based dense prediction; ARTrackV2-b256 represents discrete coordinate-sequence generation; and MCITrack-b224 represents explicit cross-frame contextual memory. (Ye et al., 2022; Bai et al., 2024; Kang et al., 2025) The three families instantiate a two-dimensional center policy, a one-dimensional coordinate-token policy, and a memory-conditioned two-dimensional center policy. We therefore report paired comparisons between every base tracker and its post-trained counterpart, emphasizing within-tracker improvements rather than cross-tracker ranking.

Training. For GOT-10k we post-train only on its training split (Huang et al., 2021), following the one-shot rule. Full-data experiments use a balanced 1:1:1 mixture of GOT-10k, LaSOT (Fan et al., 2019), and TrackingNet (Müller et al., 2018). For MCITrack, we follow the dataset composition of its original training recipe and additionally include VastTrack (Peng et al., 2024), giving a balanced 1:1:1:1 mixture. Each reset first samples a dataset uniformly and then a sequence within it, so dataset size does not change these ratios. All policies start from the corresponding author-released checkpoints. Four-GPU distributed training is used throughout: OSTrack-b256 and ARTrackV2-b256 use four RTX A6000 48-GB GPUs, whereas MCITrack-b224 uses four NVIDIA A100 80-GB GPUs. Full settings are given in Appendix C. We use AdamW and gradient clipping; supported configurations use mixed precision to reduce training cost, while probabilities, log-probabilities, and KL are always computed in fp32. (Loshchilov & Hutter, 2019; Pascanu et al., 2013; Micikevicius et al., 2018)

Inference. Base and SPOT use the upstream dataset-specific inference, evaluation, and submission code on the corresponding A6000 or A100 servers. One tracker worker is assigned to each GPU, and parallelism only partitions sequences; every sequence is still processed in temporal order by one tracker instance. The matched pair shares input resolutions, crop geometry, deterministic native readout, prediction-driven transitions, and architecture-specific state updates. During post-training, the sampled box receives the reward and centers the next crop; during evaluation, the native deterministic box drives the same transition. Thus SPOT adds neither sampling nor extra post-processing at test time, and the evaluated pair differs only in the post-trained checkpoint.

4.2 Main Results

Table 1 reports official-protocol results for each matched Base/SPOT pair.

Table 1: Base trackers and their SPOT counterparts. The ARTrackV2-b256 Base row follows the official paper, while the other Base rows are rerun from author-released models with official code and configurations (Ye et al., 2022; Bai et al., 2024; Kang et al., 2025) and may differ slightly from the original papers. GOT-10k results use the GOT-10k-only checkpoints, whereas the remaining benchmarks use the full-data checkpoints. Each Base/SPOT pair uses matched native inference; the data protocols are defined in Section 4.1.

Tracker	Variant	GOT-10k			LaSOT			$\text{LaSOT}_{\text{ext}}$			TrackingNet		
		AO	$\text{SR}_{0.5}$	$\text{SR}_{0.75}$	AUC	P_{norm}	P	AUC	P_{norm}	P	AUC	P_{norm}	P
OSTrack-b256	Base	70.5	80.0	67.8	68.5	77.1	73.9	47.2	56.8	53.0	82.6	87.3	81.4
	SPOT	**72.2**	**81.5**	**69.0**	**69.0**	**77.4**	**74.5**	**48.3**	**58.1**	**54.4**	**83.3**	**87.7**	**81.9**
ARTrackV2-b256	Base	75.9	85.4	72.7	71.6	80.2	77.2	50.8	61.9	57.7	84.9	89.3	84.5
	SPOT	**76.9**	**86.8**	**72.9**	**72.0**	**80.7**	**77.6**	**51.4**	**62.2**	**58.7**	**85.3**	**89.7**	**85.2**
MCITrack-b224	Base	77.9	88.2	76.8	75.3	85.6	83.3	54.6	65.7	62.1	86.3	90.9	86.1
	SPOT	**78.5**	**88.7**	**77.7**	**75.9**	**86.1**	**83.9**	**55.4**	**66.5**	**62.9**	**86.6**	**91.2**	**86.4**

GOT-10k. GOT-10k (Huang et al., 2021) uses a one-shot protocol in which the SPOT stage is trained only on its training split and evaluated on disjoint object classes. SPOT improves AO by 1.7, 1.0, and 0.6 points for OSTrack-b256, ARTrackV2-b256, and MCITrack-b224, respectively. The corresponding $\text{SR}_{0.75}$ gains are 1.2, 0.2, and 0.9 points, showing that sequence-level optimization improves stricter localization accuracy.

LaSOT. LaSOT (Fan et al., 2019) evaluates long-term tracking over 280 test sequences. SPOT improves AUC by 0.4–0.6 points across the three tested configurations. The largest gain is obtained by MCITrack-b224, from 75.3 to 75.9 AUC, while ARTrackV2-b256 improves from 71.6 to 72.0 AUC. These improvements are obtained without altering the original deterministic inference procedure.

TrackingNet and the LaSOT extension. On the LaSOT extension ($\text{LaSOT}_{\text{ext}}$) (Fan et al., 2021), SPOT improves AUC by 0.6–1.1 points, with an average gain of 0.8 points. TrackingNet gains are smaller, between 0.3 and 0.7 AUC points (Müller et al., 2018), but remain positive for all three tested configurations. Together with GOT-10k and LaSOT, these results show consistent gains across the tested tracker architectures, input resolutions, and benchmark protocols.

4.3 Experimental Analysis

We conduct controlled ablations primarily with OSTrack-b256, post-trained on the GOT-10k training set and evaluated on its test set. Unless otherwise stated, the ablation configuration uses $(G, T) = (16, 12)$, the trajectory return, a prediction-driven transition, head-only tuning, the default critic-free policy-gradient objective, γ=1, and β=0.04.

Sequence-level post-training comparison. We compare SPOT with SCST (self-critical sequence training), the optimization objective used in SLT, applied here to the same OSTrack-b256 policy. (Rennie et al., 2017; Kim et al., 2022) Under a matched 16-trajectory budget, 15 sampled trajectories use the shared greedy trajectory as a baseline, $A_q^{\text{SCST}} = R_q - R_{\text{greedy}}$. It does not use the group-mean baseline, global advantage normalization, or reference KL. SPOT instead uses 16 sampled trajectories, group-relative supervision, and the reference constraint. SCST and SPOT share the initial checkpoint, data, horizon, trainable CENTER head, and update budget; Base is the same checkpoint evaluated without post-training.

The Base–SCST comparison contrasts frame-level supervised training without post-training with self-critical trajectory-level post-training. The SCST–SPOT comparison captures the aggregate difference associated with group-relative advantages, global normalization, and reference KL under this controlled setting.

Table 2: Controlled post-training comparison on GOT-10k with OSTrack-b256. SCST and SPOT use the same initial checkpoint, data, rollout horizon, trainable parameters, and update budget.

Method	Optimization signal	Baseline/constraint	AO	ΔAO
Base	frame-level supervised loss	none	70.5	–
SCST	trajectory-level return	greedy trajectory	71.6	+1.1
SPOT	trajectory-level return	group mean + ref. KL	**72.2**	**+1.7**

Long-horizon performance on cumulative LaSOT prefixes. Table 2 provides a controlled comparison among Base, SCST, and SPOT using full-sequence GOT-10k AO. To examine how the benefit of SPOT varies with tracking horizon, we separately compare Base and SPOT using the standard LaSOT success AUC on cumulative temporal prefixes, without introducing a threshold-dependent drift metric. For each test sequence, we retain its first 10%, 20%, . . ., 100% of frames and recompute the success plot and AUC using the official protocol, producing ten points at 10% increments. We select one representative configuration from each tracker family: one-stream OSTrack-b256, autoregressive ARTrackV2-b256, and memory-based MCITrack-b224. Each Base/SPOT pair uses matched deterministic inference and differs only in the sequence-level post-trained checkpoint.

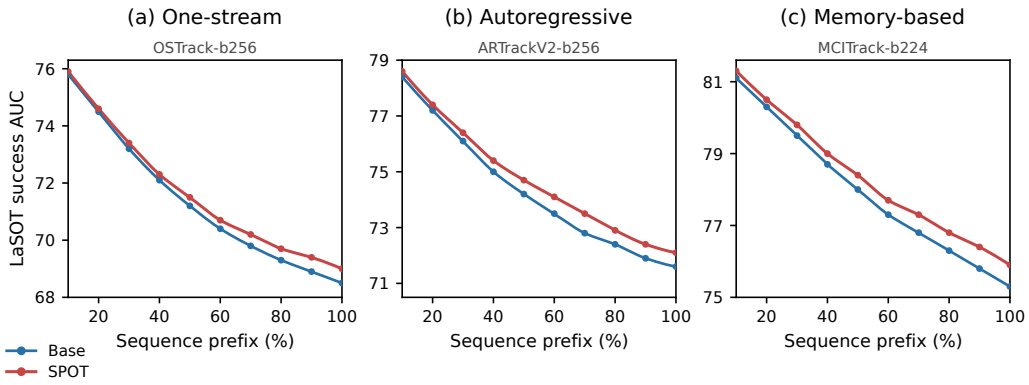

Figure 3: Long-horizon LaSOT success AUC for representative one-stream, autoregressive, and memory-based trackers. Each point recomputes the official AUC on cumulative sequence prefixes from 10% to 100% at 10% increments. Blue and red curves denote Base and SPOT, respectively. Each panel uses a tracker-specific vertical range to make its within-tracker temporal trend legible.

Across all three tracker families, SPOT outperforms the corresponding Base model at every evaluated prefix, with the separation generally becoming more pronounced toward later prefixes. This pattern is consistent with sequence-level post-training improving robustness to errors accumulated under prediction-driven state transitions. These results further demonstrate SPOT's consistent advantage over the base trackers.

Core design. Table 3 isolates two core design choices in SPOT: prediction-driven crop transitions and constrained localization-policy-only post-training. All post-trained variants in this table use the same trajectory-level return; the rows vary only the state transition and trainable parameter scope. Full SPOT reaches 72.2 AO. Re-centering crops on ground truth gives 71.2 (-1.0), and full-model tuning gives 71.5 (-0.7). These results indicate the benefit of both choices in this controlled setting.

Table 3: Core design ablation on GOT-10k with OSTrack-b256. All post-trained variants use the same trajectory-level return; rows vary the crop transition and trainable parameter scope. Blank cells indicate disabled components.

Variant	Pred. crop	Head only	AO	$SR_{0.5}$	$SR_{0.75}$
Base tracker			70.5	80.0	67.8
GT-centered crop		✓	71.2	80.4	68.3
Full fine-tuning	✓		71.5	80.7	68.4
Full SPOT	✓	✓	**72.2**	**81.5**	**69.0**

Optimization objective and reference constraint. Table 4 separately examines temporal weighting, the policy-optimization objective, and reference-KL strength. In the objective block, DPO and GRPO replace the default SPOT policy-gradient objective, while the rollout, trajectory return, policy parameterization, and AdamW update remain fixed. For DPO, each rollout group forms one preference pair from the trajectories with the highest and lowest IoU returns. Uniform weighting reaches 72.2 AO, compared with 71.0 and 71.4 for $\gamma = 0.9$ and 1.1. The default SPOT objective obtains 72.2, above GRPO (71.8) and DPO (70.7). With $\beta = 0.04$ the result is 72.2; removing the reference KL or increasing β to 0.5 reduces AO to 70.1 and 70.6, respectively.

Table 4: Temporal weighting, policy-optimization objective, and reference-KL ablations on GOT-10k with OSTrack-b256.

(a) Temporal weighting

Setting	AO	$SR_{0.5}$	$SR_{0.75}$
$\gamma = 0.9$	71.0	80.0	67.2
$\gamma = 1.0$	**72.2**	**81.5**	**69.0**
$\gamma = 1.1$	71.4	80.7	68.3

(b) Optimization method

Method	AO	$SR_{0.5}$	$SR_{0.75}$
DPO	70.7	80.0	67.9
GRPO	71.8	81.3	68.9
SPOT objective	**72.2**	**81.5**	**69.0**

(c) Reference KL

Setting	AO	$SR_{0.5}$	$SR_{0.75}$
$\beta = 0$	70.1	79.3	67.4
$\beta = 0.04$	**72.2**	**81.5**	**69.0**
$\beta = 0.5$	70.6	79.8	67.8

Sampling scale. Table 5 varies the group size and rollout length. At $T = 10$, group sizes $G = 8$, 16, and 32 yield 71.1, 71.6, and 71.3 AO, respectively. At $G = 16$, rollout lengths $T = 4$, 8, 10, and 12 yield 70.7, 71.2, 71.6, and 72.2. Thus, within this ablation grid, $(G, T) = (16, 12)$ achieves the highest AO: too few branches or too short a rollout weakens the sequence-level signal, while increasing the group beyond $G = 16$ at $T = 10$ does not improve AO.

Table 5: Group size and rollout-length ablations on GOT-10k with OSTrack-b256. $G \times T$ is the number of sampled branch steps per sequence.

Setting	AO	$SR_{0.5}$	$SR_{0.75}$
$G = 8, T = 10$	71.1	80.4	68.8
$G = 16, T = 10$	71.6	81.1	68.9
$G = 32, T = 10$	71.3	80.2	68.0
$G = 16, T = 4$	70.7	80.1	68.1
$G = 16, T = 8$	71.2	80.5	69.0
$G = 16, T = 12$	**72.2**	**81.5**	**69.0**

5 Conclusion

We presented SPOT, a sequence-level post-training framework aligning tracker optimization with prediction-driven sequential inference. SPOT optimizes native localization policies over prediction-driven trajectories with sequence-level returns, without architecture or test-time inference changes. Across three tracker families and four benchmarks, SPOT consistently improves their base trackers. Controlled analyses suggest that the gains stem from prediction-driven localization-policy optimization, highlighting the importance of training–inference state-distribution alignment. Grouped rollouts are computationally costly, limiting the scale and horizon explored under fixed compute. Future work may explore richer trajectory rewards and multi-object tracking.

AI USE STATEMENT

Generative AI tools were used for background research and literature retrieval, experimental design, drafting, translation, and language-level manuscript checking. For example, they helped identify which ablation experiments should be conducted. The authors reviewed all AI-assisted work and take responsibility for the final content of this paper.

REFERENCES

Yifan Bai, Zeyang Zhao, Yihong Gong, and Xing Wei. ARTrackV2: Prompting autoregressive tracker where to look and how to describe. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Luca Bertinetto, Jack Valmadre, Joao F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision Workshops*, pp. 850–865, 2016.

Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6182–6191, 2019.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. HIPTrack: Visual tracking with historical prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19258–19267, 2024.

Yidong Cai, Jie Liu, Jie Tang, and Gangshan Wu. Robust object modeling for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9589–9600, 2023.

Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu. Real-time actor-critic tracking. In *European Conference on Computer Vision*, pp. 318–334, 2018.

Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8126–8135, 2021.

Xin Chen, Houwen Peng, Dong Wang, Huchuan Lu, and Han Hu. Seqtrack: Sequence to sequence learning for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14572–14581, 2023.

Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6668–6677, 2020.

Yutao Cui, Cheng Jiang, Limin Wang, and Gangshan Wu. Mixformer: End-to-end tracking with iterative mixed attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13608–13618, 2022.

Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4660–4669, 2019.

Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7183–7192, 2020.

Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 518–527, 2018.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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*, 129:439–461, 2021.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 105–114, 2017.

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.

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.

Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Ben Kang, Xin Chen, Simiao Lai, Yang Liu, Yi Liu, and Dong Wang. Exploring enhanced contextual information for video-level object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4194–4202, 2025. doi: 10.1609/aaai.v39i4.32440.

Minji Kim, Seungkwan Lee, Jungseul Ok, Bohyung Han, and Minsu Cho. Towards sequence-level training for visual tracking. In *European Conference on Computer Vision*, pp. 534–551, 2022.

Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8971–8980, 2018.

Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4282–4291, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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, 2022.

Christoph Mayer, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Learning target candidate association to keep track of what not to track. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13444–13454, 2021.

Christoph Mayer, Martin Danelljan, Goutam Bhat, Matthieu Paul, Danda Pani Paudel, Fisher Yu, and Luc Van Gool. Transforming model prediction for tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8731–8740, 2022.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Matthias Müller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *European Conference on Computer Vision (ECCV)*, 2018.

Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2016.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 1310–1318, 2013.

Liang Peng, Junyuan Gao, Xinran Liu, Weihong Li, Shaohua Dong, Zhipeng Zhang, Heng Fan, and Libo Zhang. VastTrack: Vast category visual object tracking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Liangliang Ren, Xin Yuan, Jiwen Lu, Ming Yang, and Jie Zhou. Deep reinforcement learning with iterative shift for visual tracking. In *European Conference on Computer Vision*, pp. 684–700, 2018.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024, 2017.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 627–635, 2011.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1889–1897, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1683–1692, 2016.

Arnold W. M. Smeulders, Dung M. Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

Ke Song, Wei Zhang, Ran Song, and Yibin Li. Online decision based visual tracking via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

James Supancic and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr, and Bastian Leibe. Siam R-CNN: Visual tracking by re-detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6578–6588, 2020.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411–2418, 2013.

Fei Xie, Zhongdao Wang, and Chao Ma. DiffusionTrack: Point set diffusion model for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19113–19124, 2024.

Bin Yan, Houwen Peng, Jianlong Fu, Dong Wang, and Huchuan Lu. Learning spatio-temporal transformer for visual tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10448–10457, 2021a.

Bin Yan, Xinyu Zhang, Dong Wang, Huchuan Lu, and Xiaoyun Yang. Alpha-refine: Boosting tracking performance by precise bounding box estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5289–5298, 2021b.

Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *European Conference on Computer Vision (ECCV)*, 2022.

Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Dawei Zhang, Zhonglong Zheng, Riheng Jia, and Minglu Li. Visual tracking via hierarchical deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3315–3323, 2021.

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*, pp. 4010–4019, 2019.

Yaozong Zheng, Bineng Zhong, Qihua Liang, Zhiyi Mo, Shengping Zhang, and Xianxian Li. Odtrack: Online dense temporal token learning for visual tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7588–7596, 2024.

Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *European Conference on Computer Vision*, pp. 101–117, 2018.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A NATIVE-OUTPUT POLICY PARAMETERIZATION AND INSTANTIATIONS

This appendix describes how each tracker's native localization outputs are exposed through the SPOT policy interface of Eq. (1). The lightweight adapters preserve the original prediction heads and require no additional alignment stage. They define action sampling, joint log-probability, native deterministic readout, and reference KL for each localization parameterization. Appendices D and E describe the rollout environment and the training/evaluation protocol shared by all three instantiations.

Table 6: Summary of the three architecture-level instantiations of SPOT, which correspond to the three tested configurations. The RL stage is shared, while the native policy parameterization and carried state are architecture-specific. Only the listed native localization-policy parameters are trainable by default.

	OSTrack-b256	ARTrackV2-b256	MCITrack-b224
Architecture	one-stream	autoregressive	memory/SSM
Native output	CENTER	coord. tokens	CENTER
Policy geometry	2D center	four 1D coords.	2D center
Components	ctr./size/off.	l, t, r, b	ctr./size/off.
Distributions	Cat./Norm./Norm.	four Cat.	Cat./Norm./Norm.
Head replacement	none	none	none
Trainable scope	CENTER head	coord. policy	CENTER head
Carried state	none	AR feat./history	memory/SSM h
Precision	bf16/fp32	bf16/fp32	fp32

Trainable parameter budget. Table 7 reports the number of parameters enabled for gradient updates during SPOT post-training and the total parameter count of each base tracker. The trainable fraction is $100 \times N_{\text{train}}/N_{\text{total}}$. Counts exclude non-parameter buffers, such as BatchNorm running statistics, and refer to one tracker, without the frozen reference copy or optimizer state. ARTrackV2-b256 denotes the ARTrackV2-B256-Seq implementation used in our experiments; only its coordinate-token embedding and output bias are trainable. For this implementation, the embedding has 806×768 parameters and the output bias has 806, giving $619{,}814$ trainable parameters in total. The trainable scope is unchanged between the GOT-10k-only and full-data settings for each tracker. These fractions describe parameter efficiency, not the computational cost of grouped rollouts.

Throughout the paper, *head-only* is used as a concise umbrella term for restricting updates to the native localization output parameters. For ARTrackV2, this scope is slightly broader than a conventional prediction head: it also includes the coordinate-token embedding in addition to the output bias.

Table 7: Parameter budget of SPOT post-training. Trainable parameters follow the native-policy scopes in Table 6; total parameters include both trainable and frozen parameters of a single base tracker. Fractions are rounded to three decimal places.

Tracker	Trainable params.	Total params.	Trainable fraction
OSTrack-b256	6,474,117	92,518,533	6.998%
ARTrackV2-b256	619,814	134,686,247	0.460%
MCITrack-b224	4,704,645	107,153,157	4.391%

Two-dimensional CENTER policy (OSTrack-b256). OSTrack-b256 retains its native CENTER head, which produces a two-dimensional center response, a size map, and an offset map. The windowed center response is normalized over spatial cells to form a categorical center policy. At the sampled center cell, the native size prediction parameterizes a fixed-variance Normal distribution in logit space, while the native offset prediction parameterizes a fixed-variance Normal distribution in its original space. The three component log-probabilities jointly define $\log \pi_\theta(a \mid s)$. We set both continuous standard deviations to 0.15. Only the complete native CENTER head is optimized;

the backbone, Transformer, and every other non-head module remain frozen. The frozen reference is initialized from the same author-released checkpoint, and deterministic inference remains unchanged: the maximum-response center and its native size and offset are decoded into the output box.

Two-dimensional CENTER policy (MCITrack). MCITrack-b224 retains its native CENTER decoder at $224/112$ search/template resolution. It uses the same center-size-offset policy factorization as OSTrack-b256: the two-dimensional center response forms the spatial categorical component, and the size and offset predictions at the selected cell form conditional fixed-variance Normal components with standard deviation 0.15. All three components contribute to the joint log-probability and reference KL, and the complete CENTER head is optimized without replacing the decoder. All parameters outside the three CENTER decoder towers remain frozen. MCITrack additionally retains its dynamic template memory and confidence gates. In the reported MCITrack-b224 experiments, the SSM hidden state is propagated across rollout steps. The policy and frozen reference consume the same memory, hidden state, and sampled center when their KL is evaluated.

Coordinate-token vocabulary (ARTrackV2-b256). ARTrackV2-b256 is the one-dimensional instantiation of the SPOT policy interface and requires no prediction-head architectural change. It decodes a box as a sequence of coordinate tokens from a shared vocabulary. After masking the special tokens, each boundary edge is read out as an independent categorical over $K = \text{bins} \times \text{range}$ valid coordinate tokens (bins=400, range=2, i.e. K=800). The token index maps to a normalized coordinate through a fixed linear code spanning the extended domain $[-0.5, 1.5]$. In one policy forward pass the adapter obtains the four edge distributions and samples one token from each, conditioned on the same tracker state; this is the factorization in Eq. (1). Coordinates are kept in this extended domain to drive the crop transition, which preserves the tracker's original behavior, and they are clipped to $[0, 1]$ only when computing the IoU reward. The source decoder state, consisting of the decoder feature passed between frames and the coordinate history fed to the sequence model, is propagated through the rollout and replicated across the G branches when the group is first expanded. Only the native coordinate-policy parameters (the coordinate-token embedding and output bias) are optimized; all remaining parameters are frozen.

Contextual memory state (MCITrack-b224). MCITrack propagates contextual information across frames. During rollout we maintain a first-in-first-out memory bank of template and predicted-box pairs and, at each step, assemble a small set of templates for the encoder, taking a few spaced samples together with the initial template. In the reported MCITrack-b224 experiments, the contextual SSM hidden state h is propagated across rollout steps. The current policy and the frozen reference consume the same hidden state and memory context when evaluating the KL term, so Eq. (4) is measured under matched context. For numerical stability MCITrack is trained in fp32, and gradient checkpointing is disabled during the RL stage.

B Qualitative Rollout Visualization

Figure 4 shows that the positive-advantage branches ($A = +0.55$ and $A = +1.25$) maintain localization, whereas the negative-advantage branches ($A = -1.39$ and $A = -0.41$) drift at later steps. Because each branch drives its own next crop, the advantages evaluate coherent prediction-driven trajectories rather than isolated boxes.

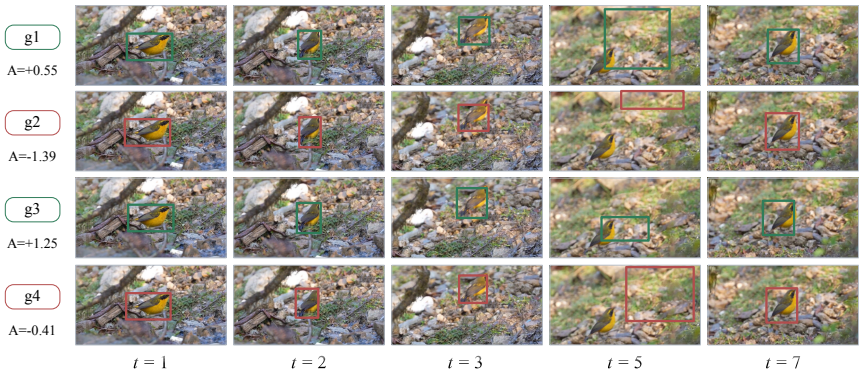

Figure 4: Grouped prediction-driven rollouts from a shared initial state. Each row is a sampled trajectory; green and red rows indicate positive and negative group-relative advantages, respectively.

C HYPERPARAMETERS

Table 8 lists the configurations used by the reported runs. B is the per-rank sequence batch size; each sequence is expanded into G independent branches. All runs use AdamW with weight decay 10^{-4}, gradient clipping at 0.1, advantage clip c=5, an advantage-std floor of 10^{-3}, and temperature 1.0. Unless otherwise stated, reported configurations use γ=1. The current policy and the frozen reference are initialized from the same author-released checkpoint, without a separate alignment stage.

Table 8: Per-tracker training configurations. Search/templ. denotes input resolutions; B is per rank, and Iter. is the number of optimizer updates per epoch. Precision lists rollout/reference-policy precision.

Tracker	search/templ.	GPUs	B	G	T	Iter.	Epochs	lr	β	Precision
OSTrack-b256	256/128	4	16	16	12	300	40	1×10^{-5}	.04	bf16/fp32
ARTrackV2-b256	256/128	4	2	8	12	1600	50	5×10^{-6}	.08	bf16/fp32
MCITrack-b224	224/112	4	4	16	8	800	30	1×10^{-5}	.04	fp32/fp32

For OSTrack and MCITrack, the complete native CENTER head is trainable and all other parameters are frozen; their size-logit and offset standard deviations are both 0.15. ARTrackV2-b256 trains the native coordinate-policy parameters while freezing the remaining tracker.

D ROLLOUT ENVIRONMENT DETAILS

Sequence sampling. Each rollout starts from a randomly selected annotated frame used only for initialization, followed by at least T annotated frames that provide rewards for the T tracking steps. Frames whose boxes are degenerate (out of image bounds or smaller than a minimum side length) are rejected with bounded retries, and a slot that cannot be filled is dropped from the batch.

Crop geometry. Templates are cropped around the annotated box of the start frame with an area factor of 2.0 and resized to the template resolution. Search regions use an area factor of 4.0. Ground-truth boxes are mapped into normalized crop coordinates for the IoU reward. The initial search region is cropped either around a jittered copy of the first-frame box (OSTrack-b256) or directly around the template box, matching each base tracker's inference initialization (ARTrackV2-b256, MCITrack).

Group expansion and transitions. At the initial state, each sequence is expanded into a group of G candidate branches. Observations, ground truth, and any tracker state (autoregressive history, template memory, SSM hidden state) are replicated per branch, after which the $B \times G$ branches

evolve independently, with each branch cropping its next search region around its *own* sampled box, exactly as at inference. Sequence starts are restricted to provide at least T rewarded transitions, so every branch has a fixed-length return.

E Training and Evaluation Protocol

Matched-input KL. For stateful trackers the KL of Eq. (4) is only meaningful if both policies see the same context. At every step the frozen reference consumes the *same* inputs (search crop, template memory, hidden state) as the current policy, and it never writes back its own state. For the boundary policy, both policies evaluate the four edge distributions under the same state s; for the CENTER policy, the conditional size and offset terms are evaluated at the same sampled center cell k. Probabilities, log-probabilities, and KL are computed in fp32 even when the rollout runs in bf16.