SRTRACK: SELF-RECOVERING VISUAL TRACKING VIA RE-DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual object tracking has made significant progress through spatio-temporal context modeling. However, current approaches lack explicit spatial priors from historical predictions to guide localization, relying on feature similarity for attention computation. More critically, existing trackers implicitly assume reliable predictions throughout the sequence and lack specific mechanisms to detect and recover from tracking failures. To address these challenges, we propose SRTrack, a self-recovering tracking framework. For failure recovery, a tracking quality estimator monitors confidence at every frame and triggers a re-localization module to predict a coarse anchor on an extended search region. For robust localization, a trajectory-guided feature sampler leverages historical positions as geometric priors for feature aggregation, and a feature refinement head performs coarse-to-fine localization through iterative deformable attention with Gaussian gates. Experiments on eight benchmarks demonstrate that SRTrack achieves competitive results, with an AO of 80.9% on GOT-10k and an AUC of 74.4% on LaSOT, surpassing the baseline HIPTrack by 3.5% and 1.7%, respectively. The code and results will be made publicly available.

1 INTRODUCTION

Visual tracking serves as a core capability in autonomous driving, surveillance, and robotics (Javed et al., 2022). It aims to localize a target throughout a video sequence, given only its bounding box in the initial frame. State-of-the-art trackers have made significant progress by modeling spatio-temporal context (Cai et al., 2024; Li et al., 2025; Xie et al., 2025) and learning robust target representations (Cai et al., 2025; Bai et al., 2024; Kang et al., 2025). However, these methods cannot explicitly assess the quality of their own predictions. When the target is lost, these trackers fail to detect their own degradation and accumulate errors without a recovery mechanism.

This problem becomes particularly severe in challenging scenarios such as fast motion, full occlusion, and out-of-view situations (Fan et al., 2019; 2021), where appearance cues become entirely unavailable. Moreover, these trackers predominantly adopt local cropping, which restricts the search region to a narrow area based on the previous prediction. As a result, trackers remain trapped in a misaligned search window, even after the target reappears (Fig. 1). This necessitates a self-recovery mechanism to recognize failure and re-detect the target in a larger search region. However, most state-of-the-art trackers (Lin et al., 2024; Guo et al., 2025; Bai et al., 2024; Cai et al., 2024) lack such mechanisms, leaving recovery from tracking failure to chance. Recently, RTracker (Huang et al., 2024) explored recoverable tracking with rule-based state switching, but it relies on a tree-structured memory bank and a re-detector MITS (Xu et al., 2023), introducing considerable computational overhead.

Aside from failure recovery, current spatio-temporal modeling remains insufficient for distractor suppression and precise localization. In the spatial dimension, existing trackers typically predict the target location through a single-stage prediction head (Ye et al., 2022; Gao et al., 2022; Cui et al., 2022), precluding progressive refinement under feature degradation. In the temporal dimension, motion modeling has shown promise for distractor suppression by providing motion cues (Zhao et al., 2025; Bai et al., 2024). However, existing approaches encode the trajectory as learnable tokens, relying on the network to implicitly discover the correspondence between coordinate values and spatial positions, rather than constructing geometric priors directly.

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.

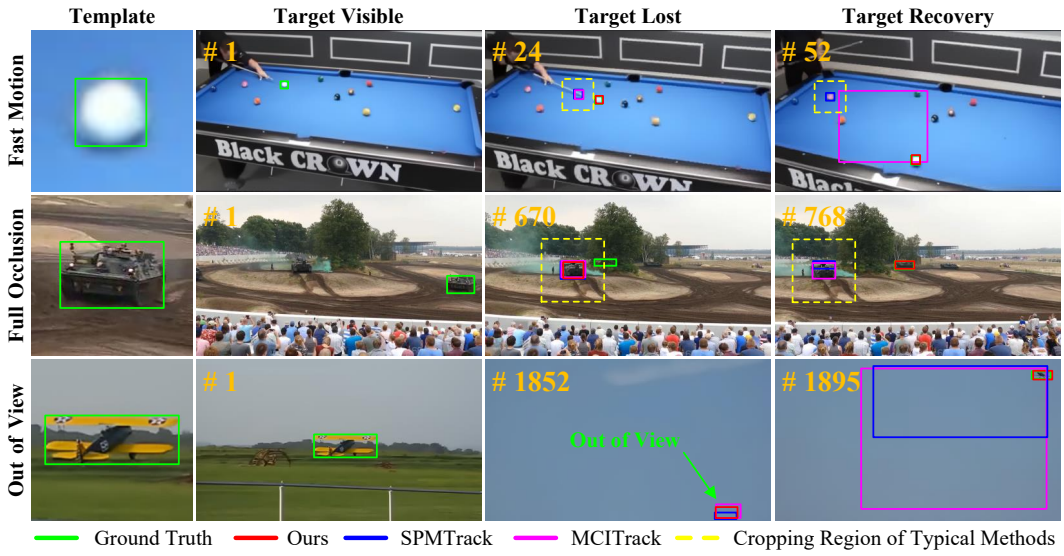

Figure 1: We compare our method with strong trackers (Cai et al., 2025; Kang et al., 2025) under challenging scenarios, including fast motion, full occlusion, and out-of-view situations. Without tracking quality estimation, existing trackers leave recovery from target loss to chance.

To address these challenges, we propose a self-recovering tracking framework termed SRTrack. For failure recovery, we introduce a tracking quality estimator that assesses prediction reliability based on appearance similarity. When failure is detected, a target re-localization module is triggered to predict a coarse anchor for re-cropping, enabling recovery from prolonged occlusions or out-of-view scenarios. For robust tracking, we introduce a trajectory-guided feature sampler that leverages historical positions as explicit geometric priors to aggregate search features, suppressing distractors under appearance ambiguity. Furthermore, a feature refinement head iteratively refines the localization across multiple prediction stages, where each stage narrows the spatial focus based on the previous one. The main contributions are summarized as follows:

- We propose SRTrack, a self-recovering tracking framework that unifies failure detection and target recovery, particularly under fast motion, full occlusion, and out-of-view scenarios.
- We introduce a trajectory-guided feature sampler that explicitly exploits historical positions as geometric priors for feature aggregation, enhancing robustness against distractors.
- We design a feature refinement head that refines localization in a coarse-to-fine manner, enabling precise tracking under feature degradation.

2 Related Work

2.1 Tracking by Re-detection

Locating the target beyond the local search region is essential for handling occlusion, fast motion, and out-of-view scenarios. Some methods address this by searching over the entire image. For example, Siam R-CNN (Voigtlaender et al., 2020) applies a re-detector at every frame and manages candidates through tracklet reasoning. CPDTrack (Zhang et al., 2024) introduces a peripheral vision branch for global awareness. More recently, SAMURAI (Yang et al., 2024) and DAM4SAM (Videnovic et al., 2025) perform full-image segmentation for tracking based on SAM2 (Ravi et al., 2025). While these methods inherently avoid excluding the target from the search region, they incur substantial computational overhead. An alternative is to retain efficient local tracking and incorporate confidence estimation. MixFormer (Cui et al., 2022) and ARTrackV2 (Bai et al., 2024) predict tracking quality scores to assess prediction reliability, but they lack effective mechanisms to recover from failure. Earlier long-term trackers such as SPLT (Yan et al., 2019) and LTMU (Dai et al., 2020) assess tracking reliability through a verifier and switch to full-image search upon detecting target absence.

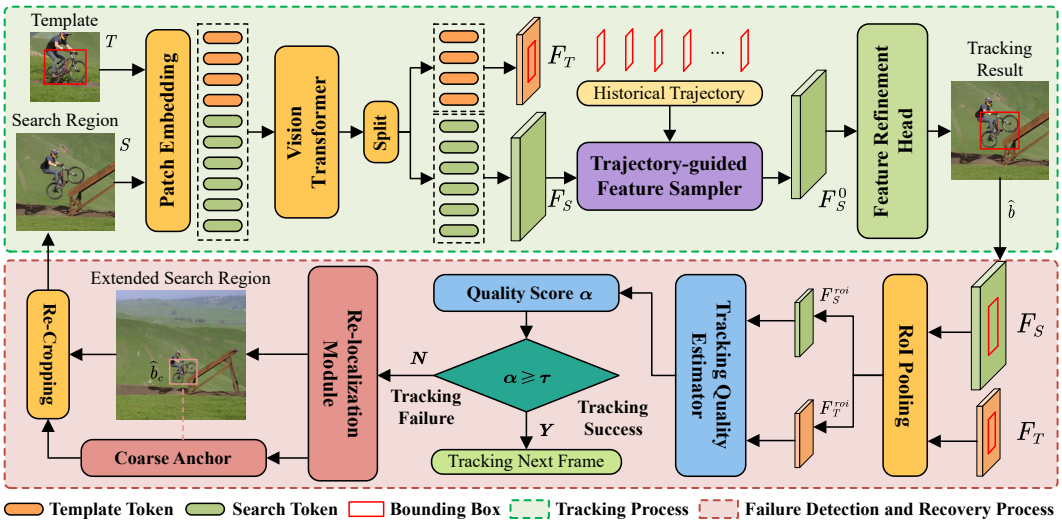

Figure 2: Overview of SRTrack. The framework consists of two processes: the tracking process (green) performs feature extraction, trajectory-guided sampling, and feature refinement for localization; the failure detection and recovery process (red) monitors tracking quality and triggers re-localization upon failure.

More recently, RTracker (Huang et al., 2024) integrates failure detection with a dedicated re-detector MITS (Xu et al., 2023) for target recovery. While these methods introduce considerable computational overhead, our approach employs a re-localization module that provides only a coarse anchor, allowing the tracker itself to perform precise localization.

2.2 Spatio-Temporal Modeling

Existing trackers employ spatio-temporal modeling to improve robustness beyond single-frame features. In the spatial dimension, most trackers predict the target location through a single-stage prediction head (Ye et al., 2022; Wei et al., 2023; Gao et al., 2022; Chen et al., 2023; Cai et al., 2023), lacking iterative refinement for precise localization. In the temporal dimension, many methods propagate dynamic templates (Shi et al., 2024; Zheng et al., 2024; Xie et al., 2025; 2024) or construct a memory bank (Cai et al., 2024; Wu et al., 2026; Videnovic et al., 2025) to learn robust target representations. For trajectory modeling, token-based approaches (Wei et al., 2023; Bai et al., 2024; Zhao et al., 2025) encode trajectory history as learnable embeddings to exploit motion cues. However, token-based methods rely on implicit learning rather than explicitly exploiting geometric priors. Instead, we directly leverage historical bounding boxes to guide spatial feature aggregation.

3 Method

3.1 Overview

As illustrated in Fig. 2, SRTrack comprises two processes: a tracking process for target localization and a failure detection and recovery process for self-recovery from tracking failures.

Tracking Process. Given a template $T \in \mathbb{R}^{3\times H_t\times W_t}$ and a search region $S \in \mathbb{R}^{3\times H_s\times W_s}$, we follow the one-stream tracking paradigm (Ye et al., 2022) to concatenate their patch embeddings and feed them into a Vision Transformer (ViT) (Dosovitskiy et al., 2021) backbone. The output tokens are split and reshaped into the template feature map $F_T \in \mathbb{R}^{C\times \frac{H_t}{p}\times \frac{W_t}{p}}$ and the search feature map $F_S \in \mathbb{R}^{C\times \frac{H_s}{p}\times \frac{W_s}{p}}$, where C is the feature dimension and p is the patch size. To leverage motion cues, the trajectory-guided feature sampler aggregates F_S using historical trajectory positions as geometric priors, producing enhanced features F_S^0. Finally, the feature refinement head takes F_S^0

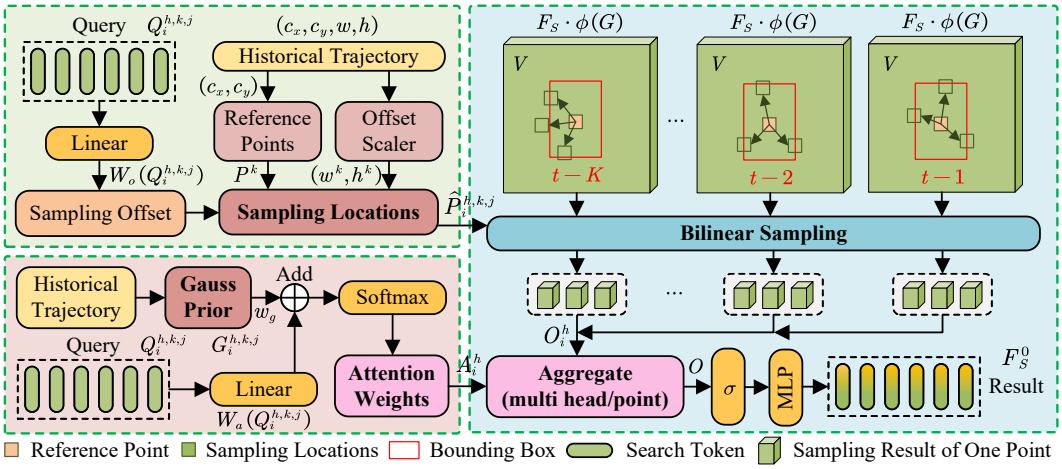

Figure 3: Illustration of the trajectory-guided feature sampling. Historical bounding boxes provide reference points and offset scale factors. The Gaussian prior biases attention weights and modulates the value features. The aggregated output is refined through sigmoid gating and an MLP.

as input and performs coarse-to-fine localization through iterative refinement with Gaussian gates, outputting the predicted bounding box $\hat{b}$.

Failure Detection and Recovery. To enable self-recovery, we introduce a tracking quality estimator that monitors tracking confidence at every frame. Specifically, we apply Region of Interest (RoI) Pooling (Jiang et al., 2018) on F_S with the predicted bounding box and on F_T with the template annotation. These pooled RoI features are fed into the tracking quality estimator, which predicts the quality score α. When $\alpha < \tau$, indicating potential tracking failure, the target re-localization module predicts a coarse anchor $\hat{b}_c$ on a larger search region. This anchor is used to re-crop a new search region, re-track the target, and obtain a second tracking result $\hat{b}_{re}$. The result with the higher quality score is selected as the final output, enabling recovery from occlusions or out-of-view scenarios.

3.2 Trajectory-Guided Feature Sampler

We propose a trajectory-guided feature sampler that leverages historical motion cues to enhance features and suppress distractors. As shown in Fig. 3, the module builds upon multi-head deformable attention (Zhu et al., 2021), replacing the learned reference points with trajectory-derived ones.

Sampling Location Prediction. Given the historical trajectory $B_{\text{traj}} = \{b^k\}_{k=1}^{K}$ from the past K frames, we extract the box centers $P^k = (c_x^k, c_y^k)$ as reference points and box sizes (w^k, h^k) as offset scale factors, where $b^k = (c_x^k, c_y^k, w^k, h^k)$ denotes the normalized bounding box. For each query position i, we compute the sampling locations by applying learned offsets:

$$\hat{P}_i^{h,k,j} = P^k + W_o(Q_i^{h,k,j}) \odot (w^k, h^k), \tag{1}$$

where $Q_i = W_q \cdot F_S$ is the query projection, $h = 1, 2, \ldots, H$ indexes the attention heads, $j = 1, 2, \ldots, J$ indexes the J sampling points per reference point, and $W_o \in \mathbb{R}^{C \times 2HKJ}$ is a learnable linear projection to predict sampling offsets.

Attention Weight Computation. To encourage sampling near the target, we compute an anisotropic Gaussian prior that measures the normalized distance from each sampling point to the box center:

$$(\Delta\hat{x}^k, \Delta\hat{y}^k) = \hat{P}_i^{h,k,j} - P^k, \quad G_i^{h,k,j} = \exp\left(-\frac{(\Delta\hat{x}^k/w^k)^2 + (\Delta\hat{y}^k/h^k)^2}{2\sigma_{attn}^2}\right), \tag{2}$$

where σ_{attn} is a learnable parameter. The attention weights are computed by incorporating the Gaussian prior into the softmax across all $K \times J$ sampling points:

$$A_i^h = \text{Softmax}_{k,j}\left(W_a(Q_i^{h,k,j}) + w_g \log G_i^{h,k,j}\right), \tag{3}$$

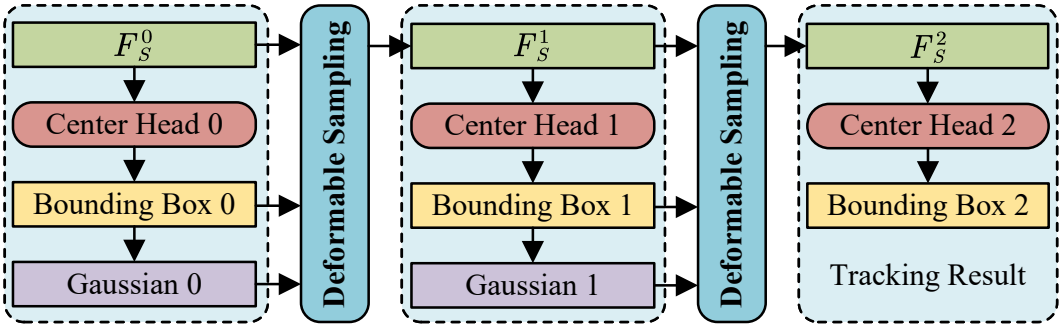

Figure 4: Feature Refinement Head. The module consists of three prediction stages. Each of the first two stages generates a Gaussian gate from its prediction to refine the feature map via deformable attention before the next stage. The final stage outputs the tracking result.

where the linear projection $W_a \in \mathbb{R}^{C\times HKJ}$ and the geometric prior strength w_g are learnable.

Feature Aggregation. Beyond adjusting the attention weights, we also modulate the value features with another Gaussian prior, integrating positional guidance into the sampled content. For each attention head h, we sample the modulated values at the predicted locations and aggregate:

$$V = F_S \odot \phi(G), \quad O_i^h = \sum_{k=1}^{K}\sum_{j=1}^{J} A_i^{h,k,j} \cdot \text{BilinearSample}(V^h, \hat{P}_i^{h,k,j}), \tag{4}$$

where ϕ is a learnable linear projection, and $V^h \in \mathbb{R}^{(C/H)\times H_s \times W_s}$ is obtained by splitting V along the channel dimension. The outputs from all heads and query positions O_i^h are concatenated to form $O \in \mathbb{R}^{N_s \times C}$. The enhanced feature is then obtained through a sigmoid gate and an MLP:

$$F_S^0 = F_S \cdot \text{Sigmoid}(\text{LN}(W_p(O))), \quad F_S^0 \leftarrow F_S^0 + \text{MLP}(\text{LN}(F_S^0)), \tag{5}$$

where W_p is the output projection, and LN denotes layer normalization.

3.3 Feature Refinement Head

We propose a feature refinement head that performs coarse-to-fine localization through iterative refinement. As shown in Fig. 4, the module consists of three prediction stages, each containing an independent center prediction head. Between consecutive stages, deformable attention with a Gaussian gate refines the feature map based on the preceding prediction.

Coarse-to-Fine Localization. Starting from the input feature map F_S^0, each stage predicts a bounding box $\hat{b}^m = (\hat{c}_x^m, \hat{c}_y^m, \hat{w}^m, \hat{h}^m)$ through its center head, where $m \in \{0, 1, 2\}$. To refine the feature map in consecutive stages, we generate an anisotropic Gaussian gate from the preceding prediction:

$$M^{l+1}(x, y) = \exp\left(-\frac{(x-\hat{c}_x^l)^2}{2\sigma_x^2} - \frac{(y-\hat{c}_y^l)^2}{2\sigma_y^2}\right), \quad (\sigma_x, \sigma_y) = (\hat{w}^l, \hat{h}^l) \cdot \sigma_f. \tag{6}$$

Here, $l \in \{0, 1\}$, (x, y) denotes coordinates on the feature map, and σ_f is a hyperparameter. The feature map is then refined through deformable attention (Zhu et al., 2021), gated by M^{l+1} to focus on the predicted region, and the result is fed into the next center head:

$$F_S^{l+1} = F_S^l + M^{l+1} \odot \text{DeformAttn}(F_S^l, \hat{b}^l), \quad \hat{b}^{l+1} = \text{CenterHead}_{l+1}(F_S^{l+1}), \tag{7}$$

where DeformAttn follows a similar mechanism in Sec. 3.2. Finally, the tracking result $\hat{b} = \hat{b}^2$.

3.4 Failure Detection and Recovery

Tracking Quality Estimator. We estimate the tracking quality by comparing the template appearance with the predicted target region. Specifically, we apply RoI Pooling (Jiang et al., 2018) on the search

Algorithm 1 Tracking with Self-Recovery

Input: Video frames $\{I_t\}_{t=1}^{N}$, initial bounding box $\hat{b}_0$, initial template T_0, failure threshold τ, update threshold τ_u
Output: Tracking results $\{\hat{b}_t\}_{t=1}^{N}$

1: $T \leftarrow T_0$ // Initialize the template
2: **for** $t = 1$ **to** N **do**
3: $S \leftarrow \text{Crop}(I_t, \hat{b}_{t-1})$ // Crop the search region
4: $\hat{b}_t, \alpha \leftarrow \text{Track}(T, S)$ // Track and estimate quality
5: **if** $\alpha \geq \tau_u$ **then**
6: $T \leftarrow \text{Crop}(I_t, \hat{b}_t)$ // Update the template
7: **if** $\alpha < \tau$ **then**
8: $S_e \leftarrow \text{Crop}_e(I_t, \hat{b}_{t-1})$ // Extend the search region
9: $\hat{b}_c \leftarrow \text{Relocalize}(T, S_e)$ // Re-localize a coarse anchor
10: $S_r \leftarrow \text{Crop}(I_t, \hat{b}_c)$ // Crop a new search region
11: $\hat{b}_{re}, \alpha_{re} \leftarrow \text{Track}(T, S_r)$ // Re-track the target
12: **if** $\alpha_{re} > \alpha$ **then**
13: $\hat{b}_t \leftarrow \hat{b}_{re}$ // Select the better tracking result
14: **return** $\{\hat{b}_t\}_{t=1}^{N}$ // Return all tracking results

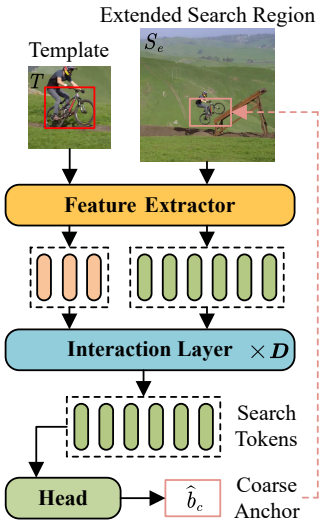

Figure 5: Illustration of the Re-localization Module.

feature F_S with the predicted bounding box $\hat{b}$, and on the template feature F_T with the template annotation b_0, obtaining F_S^{roi} and F_T^{roi}, respectively (see Figure 2). The pooled features of size $R \times R$ are reshaped into R^2 tokens of dimension C, and fed into a multi-head cross-attention layer, where F_T^{roi} serves as query and F_S^{roi} serves as key and value. The quality score is then computed as:

$$\alpha = \text{Sigmoid}(\text{MLP}(\text{CrossAttn}(F_T^{roi}, F_S^{roi}))), \tag{8}$$

where $\alpha \in [0, 1]$ is trained to estimate the IoU between the tracking result and the ground-truth.

Target Re-localization Module. When the quality score α falls below the threshold τ, we trigger the re-localization module to recover the target. As shown in Fig. 5, the module takes the template and an extended search region as input. The inputs are then independently processed by a shared feature extractor, producing template tokens and search tokens, respectively. The two token sets are concatenated and processed by D interaction layers using self-attention to enable bidirectional information exchange. Subsequently, the enhanced search tokens are reshaped into a feature map and fed into a center prediction head, which decodes the coarse anchor $\hat{b}_c$. Based on $\hat{b}_c$, the tracker re-crops a new search region and re-tracks to obtain a second prediction.

Inference Procedure. The overall inference procedure is summarized in Algorithm 1. For each frame, the tracker crops a search region based on the previous prediction and performs tracking with quality estimation (Eq. 8, Lines 3-4). When the quality score exceeds the update threshold τ_u, the dynamic template is updated with the current prediction (Lines 5-6). When the quality score falls below the failure threshold τ, an extended search region is cropped and fed into the re-localization module to predict the coarse anchor $\hat{b}_c$ (Fig. 5, Lines 8-9). A new search region is then cropped based on $\hat{b}_c$ for a second tracking process, and the result with the higher quality score between the original and recovered predictions is selected as the final tracking result (Lines 10-13).

3.5 Training Loss

We adopt a multi-stage training strategy. For the tracking process, we first train the feature refinement head, then train the trajectory-guided feature sampler. For the recovery process, we train the re-localization module and the tracking quality estimator separately. The tracking quality estimator uses a smooth L_1 loss (Girshick, 2015) for the predicted IoU score, while the training loss for all other components is:

$$\mathcal{L} = \sum_{l=0}^{L} \left(\mathcal{L}_{cls}^{l} + \lambda_{giou} \mathcal{L}_{giou}^{l} + \lambda_{L_1} \mathcal{L}_{L_1}^{l} \right), \tag{9}$$

Table 1: State-of-the-art comparison on GOT-10k, LaSOT, and TrackingNet datasets. The top two results are marked with red and blue. †Uses additional VastTrack (Peng et al., 2024) training data.

Method	GOT-10k			LaSOT			TrackingNet		
	AO(%)	$SR_{0.5}$(%)	$SR_{0.75}$(%)	AUC(%)	P_{Norm}(%)	P(%)	AUC(%)	P_{Norm}(%)	P(%)
Ours	**80.9**	**91.4**	**79.7**	74.4	**84.5**	82.1	85.2	90.3	85.1
Δ (vs HIPTrack)	(+3.5)	(+3.4)	(+5.2)	(+1.7)	(+1.6)	(+2.6)	(+0.7)	(+1.2)	(+1.3)
HIPTrack (Cai et al., 2024)	77.4	88.0	74.5	72.7	82.9	79.5	84.5	89.1	83.8
SPMTrack-L (Cai et al., 2025)	**80.0**	**89.4**	**79.9**	**76.8**	**85.9**	**84.0**	**86.9**	**91.0**	**87.2**
MCITrack-L† (Kang et al., 2025)	**80.0**	**88.5**	**80.2**	**76.6**	**86.1**	**85.0**	**87.9**	**92.1**	**89.2**
ARTrackV2-L384 (Bai et al., 2024)	**79.5**	87.8	79.6	73.6	82.8	81.1	**86.1**	**90.4**	86.2
ARTrackL384 (Wei et al., 2023)	78.5	87.4	77.8	73.1	82.2	80.3	85.6	89.6	86.0
ODTrack-L (Zheng et al., 2024)	78.2	87.2	77.3	74.0	84.2	**82.3**	**86.1**	**91.0**	**86.7**
RTracker-L (Huang et al., 2024)	77.9	87.0	76.9	74.7	**84.5**	-	-	-	-
LoRAT-L (Lin et al., 2024)	77.5	86.2	78.1	**75.1**	84.1	82.0	85.6	89.7	85.4
EVPTrack-384 (Shi et al., 2024)	76.6	86.7	73.9	72.7	82.9	80.3	84.4	89.1	-
MambaLCT-384 (Li et al., 2025)	76.2	86.7	74.3	73.6	84.1	81.6	85.2	89.8	85.2
AQATrack-384 (Xie et al., 2024)	76.0	85.2	74.9	72.7	82.9	80.2	84.8	89.3	84.3
DropTrack (Wu et al., 2023)	75.9	86.8	72.0	71.8	81.8	78.1	84.1	88.9	-
SeqTrack-L384 (Chen et al., 2023)	74.8	81.9	72.2	72.5	81.5	79.3	85.5	89.8	85.8
ROMTrack-384 (Cai et al., 2023)	74.2	84.3	72.4	71.4	81.4	78.2	84.1	89.0	83.7
TATrack-L (He et al., 2023)	-	-	-	71.1	79.1	76.1	85.0	89.3	84.5
GRM (Gao et al., 2023)	73.4	82.9	70.4	69.9	79.3	75.8	84.0	88.7	83.3
CTTrack-L (Song et al., 2023)	72.8	81.3	71.5	69.8	79.7	76.2	84.9	89.1	83.5

where L is the index of the last prediction stage, and $\mathcal{L}_{cls}^{l}$ denotes the focal loss (Lin et al., 2017) for classification. The regression loss consists of GIoU loss (Rezatofighi et al., 2019) $\mathcal{L}_{giou}^{l}$ and L_1 loss $\mathcal{L}_{L_1}^{l}$, where $\lambda_{giou} = 2.0$ and $\lambda_{L_1} = 5.0$, following the default setting in HIPTrack (Cai et al., 2024).

4 Experiments

4.1 Implementation Details

Model Settings. SRTrack employs a frozen ViT-L (Dosovitskiy et al., 2021) as the backbone. The template and search region are resized to 192×192 and 384×384 pixels, respectively. The trajectory-guided feature sampler uses $H = 8$ attention heads, $K = 8$ historical frames, and $J = 4$ sampling points per reference point. The feature refinement head sets $\sigma_f = 1.0$ for the Gaussian gate. The tracking quality estimator uses RoI pooling of size $R = 7$. The re-localization module employs a frozen DINOv2 (Oquab et al., 2023) ViT-S/14 backbone with $D = 4$ interaction layers.

Training and Inference. We train SRTrack on the training splits of COCO (Lin et al., 2014), GOT-10k (Huang et al., 2019), LaSOT (Fan et al., 2019), and TrackingNet (Muller et al., 2018). All stages use the AdamW optimizer (Loshchilov & Hutter, 2019) with linear warm-up and cosine annealing. The maximum and minimum learning rates are 3×10^{-4} and 1×10^{-5}, respectively. For the tracking process, we first train the feature refinement head for 100 epochs with a batch size of 6, then train the trajectory-guided feature sampler for the same number of epochs and batch size. For the recovery process, we train the re-localization module for 50 epochs with a batch size of 20, and train the tracking quality estimator for 50 epochs with a batch size of 6. During inference, the template update threshold τ_u is set to 0.8, while the failure threshold τ is set between 0.55 and 0.65 depending on the dataset. For GOT-10k inference, SRTrack runs at 24.6 FPS on an NVIDIA RTX 4070Ti GPU.

4.2 Comparisons with the State-of-the-art

GOT-10k (Huang et al., 2019) contains 180 test sequences. For a fair comparison, we train our method only on the GOT-10k training split. SRTrack achieves an AO of 80.9% on GOT-10k, outperforming the baseline HIPTrack (Cai et al., 2024) by 3.5%.

LaSOT (Fan et al., 2019) and **LaSOT$_{\text{ext}}$** (Fan et al., 2021) are large long-term tracking benchmarks. SRTrack achieves 74.4% AUC on LaSOT (Table 1), surpassing HIPTrack by 1.7%. As shown in Figure 6, SRTrack achieves competitive performance in out-of-view and fast-motion scenarios, while remaining limited under viewpoint change. SRTrack achieves 54.0% AUC on LaSOT$_{\text{ext}}$ (Table 2), surpassing the baseline HIPTrack by 1.0%.

Table 2: State-of-the-art comparison on $\text{LaSOT}_{\text{ext}}$, TNL2K, OTB2015, UAV123, and NFS30. The top two results are marked with red and blue.

(a) $\text{LaSOT}_{\text{ext}}$

Method	AUC(%)	P_{Norm}(%)	P(%)
Ours	**54.0**	**65.5**	**62.1**
MCITrack-L† (Kang et al., 2025)	**55.7**	**66.5**	**62.9**
RTracker-L (Huang et al., 2024)	**54.9**	**65.5**	**62.7**
ODTrack-L (Zheng et al., 2024)	53.9	**65.4**	61.7
EVPTrack-384 (Shi et al., 2024)	53.7	**65.5**	61.9
TemTrack (Xie et al., 2025)	53.4	64.8	61.0
ARTrackV2-L384 (Bai et al., 2024)	53.4	63.7	60.2
MambaLCT-384 (Li et al., 2025)	53.3	64.8	61.4
HIPTrack (Cai et al., 2024)	53.0	64.3	60.6

(b) TNL2K

Method	AUC(%)
Ours	61.4
SPMTrack-L (Cai et al., 2025)	**63.7**
LoRAT-L (Lin et al., 2024)	**62.3**
ODTrack-L (Zheng et al., 2024)	**61.7**
ARTrackV2-L384 (Bai et al., 2024)	61.6
ARTrackL384 (Wei et al., 2023)	60.3
EVPTrack-384 (Shi et al., 2024)	59.1
TemTrack (Xie et al., 2025)	58.8
MambaLCT-384 (Li et al., 2025)	58.5

(c) OTB2015

Method	AUC(%)
Ours	**72.7**
ODTrack-L (Zheng et al., 2024)	**72.4**
EVPTrack-384 (Shi et al., 2024)	**72.3**
HIPTrack (Cai et al., 2024)	71.0
ROMTrack-384 (Cai et al., 2023)	70.9
MixViT-L (Cui et al., 2022)	70.7
DropTrack (Wu et al., 2023)	69.6
AiATrack (Gao et al., 2022)	69.6

(d) UAV123

Method	AUC(%)
Ours	**71.4**
MCITrack-L† (Kang et al., 2025)	**71.5**
CTTrack-L (Song et al., 2023)	**71.3**
ARTrackL384 (Wei et al., 2023)	71.2
TemTrack (Xie et al., 2025)	70.8
HIPTrack (Cai et al., 2024)	70.5
MixViT-L (Cui et al., 2022)	68.7
SeqTrack-L384 (Chen et al., 2023)	68.5

(e) NFS30

Method	AUC(%)
Ours	**68.9**
ROMTrack-384 (Cai et al., 2023)	**68.8**
ARTrackV2-L384 (Bai et al., 2024)	**68.4**
HIPTrack (Cai et al., 2024)	68.1
ARTrackL384 (Wei et al., 2023)	67.9
AiATrack (Gao et al., 2022)	67.9
OSTrack384 (Ye et al., 2022)	66.5
SeqTrack-L384 (Chen et al., 2023)	66.2

Table 3: Ablation analysis of proposed components. FRH: feature refinement head. TFS: trajectory-guided feature sampler. SR: self-recovering module (tracking quality estimator and re-localization module).

#	Backbone	FRH	TFS	SR	GOT-10k AO(%)	LaSOT AUC(%)	UAV123 AUC(%)
1	ViT-B				77.4	72.7	69.8
2	ViT-L				78.4 (+1.0)	73.0 (+0.3)	70.0 (+0.2)
3	ViT-L	✓			79.7 (+1.3)	73.9 (+0.9)	70.4 (+0.4)
4	ViT-L	✓	✓		80.6 (+0.9)	74.4 (+0.5)	70.9 (+0.5)
5	ViT-L	✓	✓	✓	**80.9** (+0.3)	**74.4** (+0.0)	**71.4** (+0.5)

Table 4: Ablation analysis of trajectory length K and training clip length δ on GOT-10k. The default $K=8$, $\delta=19$.

(a) Trajectory Length			(b) Clip Length		
K	δ	AO(%)	K	δ	AO(%)
2	19	**80.7**	8	11	79.9
4	19	80.1	8	15	80.1
8	19	80.6	8	19	**80.6**
12	19	79.8	8	23	80.3
16	19	79.7	8	27	79.8

TrackingNet (Muller et al., 2018) and **TNL2K** (Wang et al., 2021) are challenging benchmarks across diverse scenarios. On TrackingNet and TNL2K, SRTrack achieves 85.2% and 61.4% AUC, respectively, improving over HIPTrack by 0.7% on TrackingNet.

OTB2015 (Wu et al., 2015), **UAV123** (Mueller et al., 2016), and **NFS30** (Kiani Galoogahi et al., 2017) are classical benchmarks. In Table 2, SRTrack achieves 72.7% AUC on OTB2015, 71.4% on UAV123, and 68.9% on NFS30, outperforming HIPTrack by 1.7%, 0.9%, and 0.8%, respectively.

4.3 Ablation Studies

Component Analysis. As shown in Table 3, we progressively add each component to isolate its contribution. FRH and TFS bring consistent improvements across all three benchmarks. SR further improves GOT-10k AO by 0.3% and UAV123 AUC by 0.5%, while maintaining LaSOT performance.

Effect of Trajectory Length. As shown in Table 4(a), the performance peaks at $K=2$ and $K=8$. Extending beyond 8 frames degrades performance, with $K=16$ dropping by 0.9%, as distant positions introduce noise from outdated motion. We adopt $K=8$ as the default to balance sufficient motion context against noise from outdated positions.

Effect of Training Clip Length. As shown in Table 4(b), we vary the number of consecutive frames δ sampled per training clip while fixing $K=8$. A moderate $\delta=19$ achieves the best result. Shorter clips limit the temporal diversity of training trajectories, while longer clips introduce sparser and less coherent historical positions, leading to degraded performance. We adopt $\delta=19$ as the default.

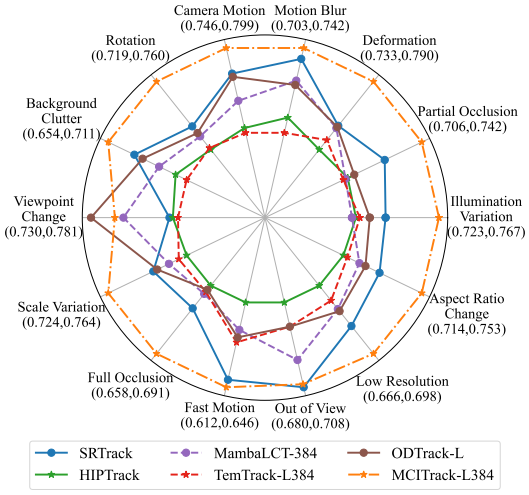

Figure 6: Attribute-wise comparison on LaSOT.

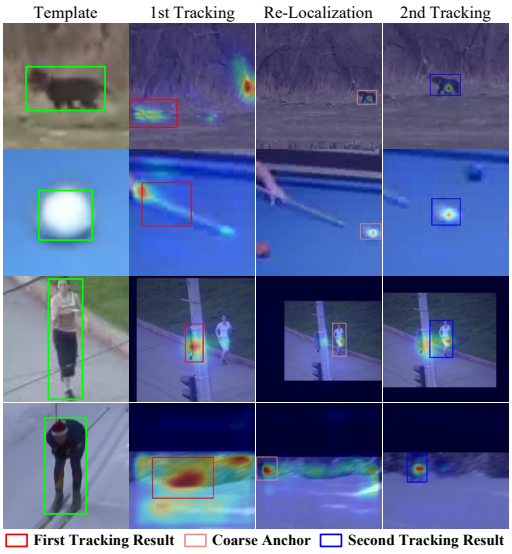

Figure 7: Visualization of the tracking recovery.

Table 5: Comparison of the re-localization module.

Methods	Trainable Params
MITS	32.8M
Ours	**10.9M**

Table 6: Ablation of the re-localization module design on GOT-10k.

Feature Extractor	Interaction	Spatial Prior	AO(%)
ResNet-18 (Stage 1–3)	Cross-attn	✓	39.1
DINOv2 (ViT-S/14)	Cross-attn	✓	58.5
DINOv2 (ViT-S/14)	Cross-attn		59.5
DINOv2 (ViT-S/14)	Self-attn		**63.9**
ViT-L (Shared)	Self-attn		57.9

Effect of the Re-localization Module. To evaluate the module's capability, we directly use its prediction as the final tracking result. As shown in Table 6, our default design achieves the highest AO of 63.9%. Replacing cross-attention with self-attention improves AO (Lines 3 and 4). Adding a spatial prior derived from the previous prediction does not improve performance (Lines 2 vs 3), since the prior can bias feature matching toward the center of the search region. Sharing the tracker's ViT-L backbone causes a 6.0% AO drop (Lines 4 and 5). We attribute this to an architectural mismatch: ViT-L is trained as a one-stream joint encoder where template and search tokens interact at every layer, while the re-localization module requires encoding each image independently before interaction. Notably, our module uses 3× fewer trainable parameters than MITS (Xu et al., 2023) (Table 5).

Visualization of Tracking Recovery. Figure 7 illustrates the self-recovery process across different scenarios. When the target is lost due to fast motion or occlusion, the first tracking shows misaligned responses (column 2). The tracking quality estimator detects the degradation and triggers re-localization, which predicts a coarse anchor on the extended search region (column 3). Based on this anchor, the tracker recovers to the correct target in the second tracking (column 4).

5 Conclusion

We presented SRTrack, a self-recovering visual tracking framework that unifies failure recovery and robust localization. The framework monitors tracking quality at every frame and triggers re-localization upon failure, while leveraging trajectory-guided feature sampling and a feature refinement head for precise localization. Experiments on eight datasets demonstrate competitive performance, with an AO of 80.9% on GOT-10k and an AUC of 74.4% on LaSOT, while running at 24.6 FPS. Our current failure detection relies on a single confidence threshold, and the re-localization module provides only a coarse anchor, leaving room for more adaptive detection and stronger recovery.

AI USE STATEMENT

During the preparation of this work, we used generative AI tools in order to draft parts of the paper, improve language readability, and assist with code implementation. After using AI tools, we reviewed and edited the content as needed and take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

We provide implementation details including model architecture, training hyperparameters, and inference settings in Section 4.1, and describe the multi-stage training procedure in Section 3.5. All experiments use publicly available benchmarks with the corresponding standard evaluation protocols. Source code and pretrained models will be released upon acceptance.

REFERENCES

Yifan Bai, Zeyang Zhao, Yihong Gong, and Xing Wei. Artrackv2: Prompting autoregressive tracker where to look and how to describe. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19048–19057, 2024.

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.

Wenrui Cai, Qingjie Liu, and Yunhong Wang. Spmtrack: Spatio-temporal parameter-efficient fine-tuning with mixture of experts for scalable visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16871–16881, 2025.

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.

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.

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.

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. IEEE, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*, pp. 1–22, 2021.

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*, pp. 5374–5383, 2019.

Heng Fan, Hexin Bai, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Harshit, Mingzhen Huang, Juehuan Liu, et al. Lasot: A high-quality large-scale single object tracking benchmark. *International Journal of Computer Vision*, 129(2):439–461, 2021.

Shenyuan Gao, Chunluan Zhou, Chao Ma, Xinggang Wang, and Junsong Yuan. Aiatrack: Attention in attention for transformer visual tracking. In *Proceedings of the European Conference on Computer Vision*, pp. 146–164, 2022.

Shenyuan Gao, Chunluan Zhou, and Jun Zhang. Generalized relation modeling for transformer tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18686–18695, 2023.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

Mingzhe Guo, Weiping Tan, Wenyu Ran, Liping Jing, and Zhipeng Zhang. Dreamtrack: Dreaming the future for multimodal visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7210, 2025.

Kaijie He, Canlong Zhang, Sheng Xie, Zhixin Li, and Zhiwen Wang. Target-aware tracking with long-term context attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 773–780, 2023.

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, 2019.

Yuqing Huang, Xin Li, Zikun Zhou, Yaowei Wang, Zhenyu He, and Ming-Hsuan Yang. Rtracker: Recoverable tracking via pn tree structured memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19038–19047, 2024.

Yuqing Huang, Liting Lin, Weijun Zhuang, Zhenyu He, and Xin Li. Drift-resilient temporal priors for visual tracking. *arXiv preprint arXiv:2604.02654*, 2026.

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, 2022.

Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision*, pp. 784–799, 2018.

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.

Hamed Kiani Galoogahi, Ashton Fagg, Chen Huang, Deva Ramanan, and Simon Lucey. Need for speed: A benchmark for higher frame rate object tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1125–1134, 2017.

Xiaohai Li, Bineng Zhong, Qihua Liang, Guorong Li, Zhiyi Mo, and Shuxiang Song. Mambalct: Boosting tracking via long-term context state space model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4986–4994, 2025.

Liting Lin, Heng Fan, Zhipeng Zhang, Yaowei Wang, Yong Xu, and Haibin Ling. Tracking meets lora: Faster training, larger model, stronger performance. In *Proceedings of the European Conference on Computer Vision*, pp. 300–318, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2980–2988, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, pp. 1–8, 2019.

Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Proceedings of the European Conference on Computer Vision*, pp. 445–461, 2016.

Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision*, pp. 300–317, 2018.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint*, 2023.

Liang Peng, Junyuan Gao, Xinran Liu, Weihong Li, Shaohua Dong, Zhipeng Zhang, Heng Fan, and Libo Zhang. Vasttrack: Vast category visual object tracking. *Advances in Neural Information Processing Systems*, 37:130797–130818, 2024.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *Proceedings of the International Conference on Learning Representations*, pp. 1–42, 2025.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 658–666, 2019.

Liangtao Shi, Bineng Zhong, Qihua Liang, Ning Li, Shengping Zhang, and Xianxian Li. Explicit visual prompts for visual object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 4838–4846, 2024.

Zikai Song, Run Luo, Junqing Yu, Yi-Ping Phoebe Chen, and Wei Yang. Compact transformer tracker with correlative masked modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2321–2329, 2023.

Jovana Videnovic, Alan Lukezic, and Matej Kristan. A distractor-aware memory for visual object tracking with sam2. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24255–24264, 2025.

Paul Voigtlaender, Jonathon Luiten, Philip HS 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.

Xiao Wang, Xiujun Shu, Zhipeng Zhang, Bo Jiang, Yaowei Wang, Yonghong Tian, and Feng Wu. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13763–13773, 2021.

Xing Wei, Yifan Bai, Yongchao Zheng, Dahu Shi, and Yihong Gong. Autoregressive visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9697–9706, 2023.

Qiangqiang Wu, Tianyu Yang, Ziquan Liu, Baoyuan Wu, Ying Shan, and Antoni B. Chan. Dropmae: Masked autoencoders with spatial-attention dropout for tracking tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14561–14571, 2023.

Weijing Wu, Qihua Liang, Bineng Zhong, Xiaohu Tang, Yufei Tan, Ning Li, and Yuanliang Xue. Mutrack: A memory-aware unified representation framework for visual tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 10772–10780, 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.

Jinxia Xie, Bineng Zhong, Zhiyi Mo, Shengping Zhang, Liangtao Shi, Shuxiang Song, and Rongrong Ji. Autoregressive queries for adaptive tracking with spatio-temporal transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19300–19309, 2024.

Jinxia Xie, Bineng Zhong, Qihua Liang, Ning Li, Zhiyi Mo, and Shuxiang Song. Robust tracking via mamba-based context-aware token learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 8727–8735, 2025.

Yuanyou Xu, Zongxin Yang, and Yi Yang. Integrating boxes and masks: A multi-object framework for unified visual tracking and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9738–9751, 2023.

Bin Yan, Haojie Zhao, Dong Wang, Huchuan Lu, and Xiaoyun Yang. 'skimming-perusal'tracking: A framework for real-time and robust long-term tracking. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2385–2393. IEEE, 2019.

Cheng-Yen Yang, Hsiang-Wei Huang, Wenhao Chai, Zhongyu Jiang, and Jenq-Neng Hwang. Samurai: Adapting segment anything model for zero-shot visual tracking with motion-aware memory. *arXiv preprint*, 2024.

Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In *Proceedings of the European Conference on Computer Vision*, pp. 341–357, 2022.

Dailing Zhang, Shiyu Hu, Xiaokun Feng, Xuchen Li, Jing Zhang, Kaiqi Huang, et al. Beyond accuracy: Tracking more like human via visual search. *Advances in Neural Information Processing Systems*, 37:2629–2662, 2024.

Jie Zhao, Xin Chen, Yongsheng Yuan, Michael Felsberg, Dong Wang, and Huchuan Lu. Efficient motion prompt learning for robust visual tracking. In *Proceedings of the International Conference on Machine Learning*, pp. 1–18, 2025.

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.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations*, pp. 1–16, 2021.

A SUPPLEMENTARY MATERIAL

A.1 ADDITIONAL ABLATION STUDIES

Table A1: Ablation analysis of the trajectory-guided feature sampler on GOT-10k. Gaussian Prior: whether the Gaussian prior biases attention scores. Sampled Feature: features being sampled, where F_S, G, and ϕ denote search features, Gaussian spatial maps, and a learnable projection, respectively.

#	Feature Sampling Strategy	Gaussian Prior	Sampled Feature (Value)	GOT-10k AO(%)
1	no sampling		F_S	79.7
2	Deformable (9 points)		F_S	79.7
3	Deformable (box center)		F_S	78.6
4	Deformable (box center)	✓	F_S	80.0
5	Cross-Attn		G	80.4
6	Deformable (box center)	✓	$F_S \odot \phi(G)$	**80.6**

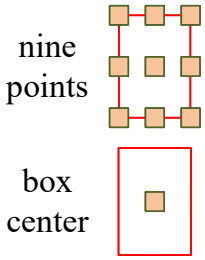

Figure A1: Reference point configurations in deformable sampling.

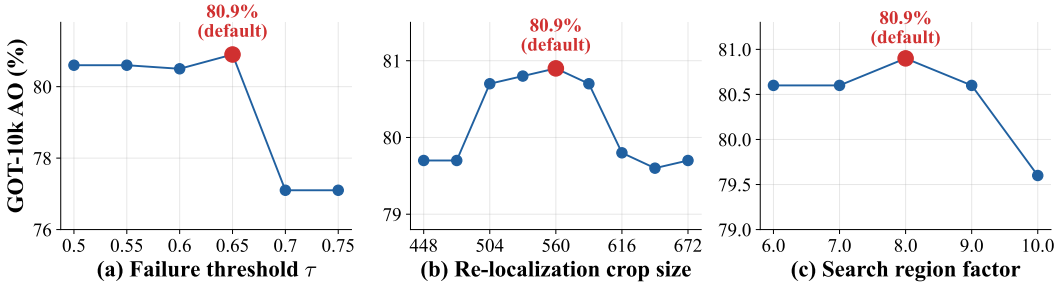

Figure A2: Sensitivity analysis of three hyperparameters on GOT-10k. (a) Failure threshold τ. (b) Re-localization crop size. (c) Search region factor. Red dots indicate our default settings.

Design of the Trajectory-Guided Feature Sampler. We investigate the design choices of feature sampling strategy, Gaussian prior, and sampled feature in Table A1. The baseline without the trajectory module achieves 79.7% AO (Line 1). Deformable sampling with nine reference points covering the bounding box (Line 2, Fig. A1) shows no gain over the baseline. Moreover, using only the box center as the reference point degrades performance by 1.1% (Line 3), as a single reference point provides insufficient geometric guidance for the sampling offsets. However, adding the Gaussian prior as a bias on the attention scores (Line 4) constrains the sampling offsets toward the target region, improving AO by 1.4% over Line 3. Furthermore, directly sampling from the Gaussian spatial maps via cross-attention (Line 5) yields a 0.7% gain over the baseline, demonstrating the effectiveness of incorporating spatial guidance into the sampled content. Our final design (Line 6) integrates the Gaussian prior into both the attention weights and the value features, achieving the best AO of 80.6%.

Hyperparameter Sensitivity. Fig. A2 analyzes three hyperparameters of the self-recovery module on GOT-10k. The failure threshold τ (Fig. A2a) is the most sensitive, with performance peaking at $\tau = 0.65$ and declining to 77.1% at $\tau = 0.70$. This sharp drop occurs because a higher threshold triggers re-localization too frequently, and the re-localization module only provides coarse anchors with limited precision. In contrast, the crop size (Fig. A2b) and search region factor (Fig. A2c) are less sensitive, with stable performance around the default values. We additionally evaluate the sensitivity of τ and crop size on UAV123 (Mueller et al., 2016) (Tables A2 and A3). The failure threshold τ shows the same trend as on GOT-10k: performance peaks in the 0.57–0.59 range and degrades at higher values due to excessive triggering. The crop size remains stable across the full sweep, with a variation of only 0.23% AUC, confirming that these trends are consistent across datasets.

Attribute Analysis on UAV123. Fig. A3 presents the per-attribute success plots on UAV123. SRTrack ranks first in eleven of twelve attributes, with notable gains under out-of-view and full occlusion,

Table A2: Sensitivity of τ on UAV123.

τ	0.50	0.55	0.57	0.59	0.60	0.65
AUC(%)	70.87	70.93	71.32	**71.33**	71.25	70.52

Table A3: Sensitivity of crop size on UAV123.

Crop size	392	420	448	476	504	532	560	588
AUC(%)	71.33	**71.37**	71.29	71.15	71.33	71.21	71.29	71.30

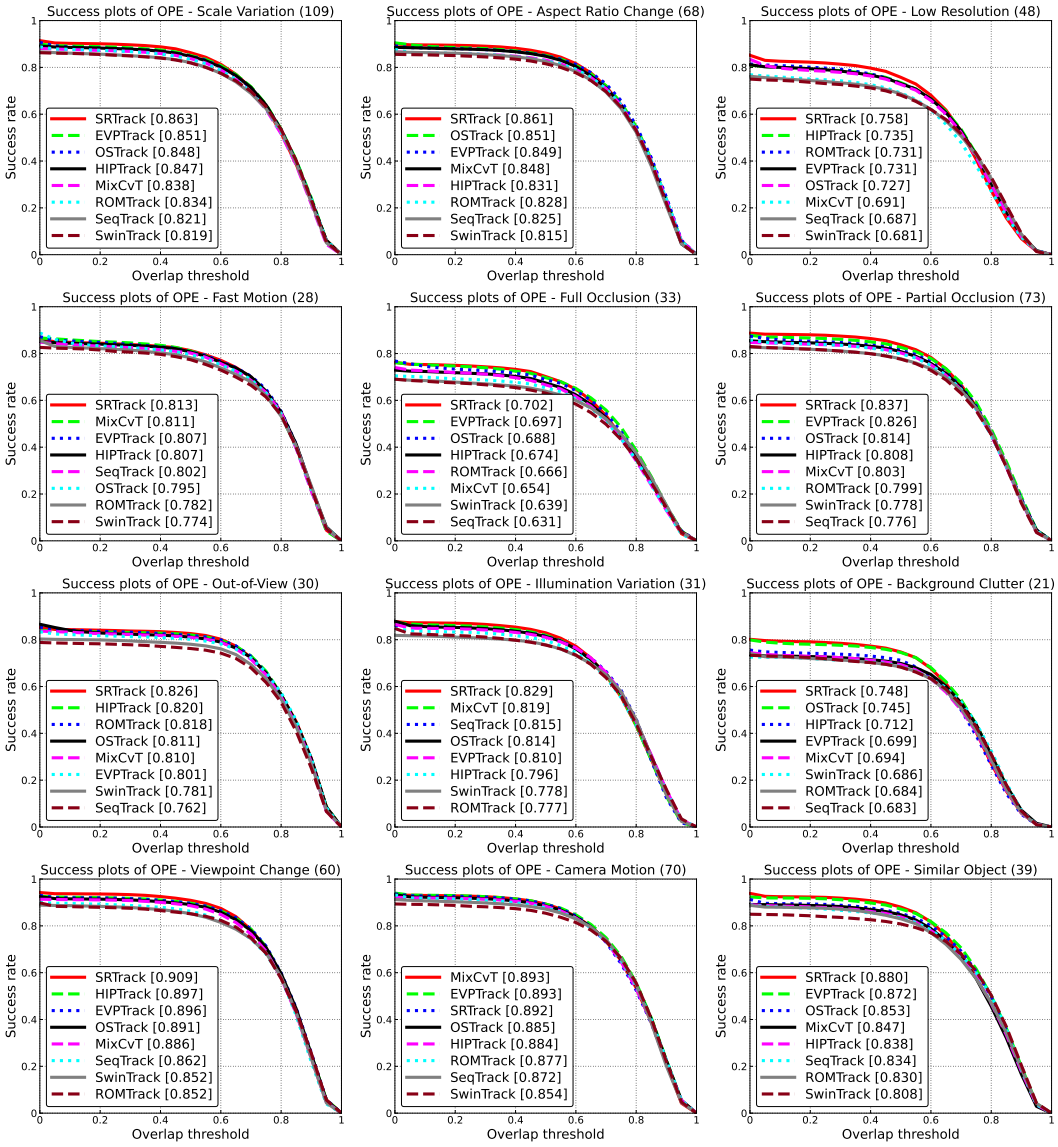

Figure A3: Success plots on UAV123 across 12 challenging factors. The numbers in brackets denote $SR_{0.5}$ scores (success rates at an overlap threshold of 0.5).

where the self-recovery mechanism helps relocate the target after disappearance. Performance is relatively weaker under camera motion (ranked 3rd), where abrupt camera movement shifts the target's position, reducing the reliability of trajectory-based priors.

Computational Efficiency. Table A4 compares the computational cost of SRTrack with recent trackers of comparable accuracy. SRTrack processes only one template and one search region per frame through the backbone, whereas SPMTrack-L and DTPTrack-L (Huang et al., 2026) feed multiple reference frames and result in higher MACs. The tracking quality estimator adds only 0.26 G MACs per frame. Re-localization adds 70 G MACs plus one additional tracking pass, but is triggered only upon detected failure. The reported FPS of 24.6 on GOT-10k already includes this overhead.

Table A4: Computational efficiency comparison on GOT-10k. The MACs are hardware-independent.

Method	Backbone Input	AO(%)	MACs (G)	FPS
SRTrack (Ours)	1 template + 1 search frame	**80.9**	**238**	24.6[a]
DTPTrack-L378 (Huang et al., 2026)	4 templates + 1 search frame	80.3	581	6.1[b]
SPMTrack-L (Cai et al., 2025)	3 templates + 1 search frame	80.0	975	5.0[b]

[a] RTX 4070Ti. [b] A100 (from DTPTrack).

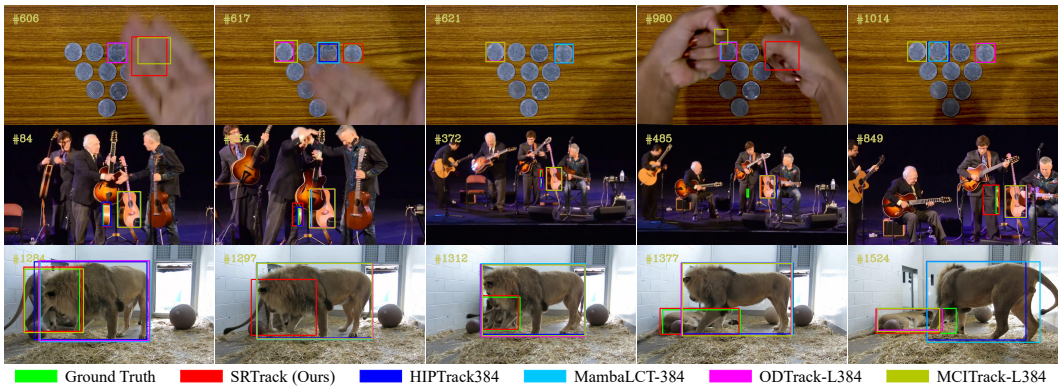

Figure A4: Qualitative comparison under full occlusion on LaSOT. From top to bottom: *coin-3*, *guitar-10*, and *lion-12*.



Figure A5: Qualitative comparison under out-of-view scenarios on LaSOT. From top to bottom: *lizard-6*, *robot-5*, and *yoyo-17*.

A.2 Additional Qualitative Results

Visualization on LaSOT. We present qualitative comparisons under three challenging scenarios. Under full occlusion (Fig. A4), SRTrack maintains localization on the correct target throughout the occlusion period. After the occlusion ends, HIPTrack (Cai et al., 2024) and MambaLCT (Li et al., 2025) drift to neighboring distractors with similar appearance, while SRTrack successfully recovers to the original target. Under out-of-view (Fig. A5), SRTrack re-localizes the target after it leaves and re-enters the field of view. HIPTrack and ODTrack (Zheng et al., 2024), lacking an explicit recovery mechanism, drift to background regions or nearby distractors once the target disappears and fail to re-acquire it upon re-entry. Under fast motion (Fig. A6), SRTrack recovers the target after large inter-frame displacements. MambaLCT and MCITrack (Kang et al., 2025) lose the target once it moves beyond the local search region and drift to other objects for the remainder of the sequence.

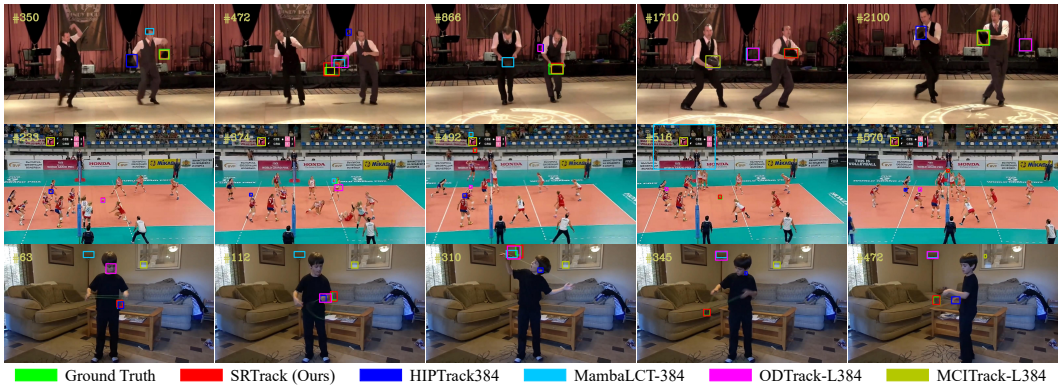

Figure A6: Qualitative comparison under fast motion on LaSOT. From top to bottom: *hat-1*, *volleyball-19*, and *yoyo-7*.