Motion-aware Sparse Pipeline for Lightweight Object Tracking

Qingmao Wei[1,2], Fagui Liu[1,2]✉, Dengke Zhang[1], Qingze He[1], and Quan Tang[2]✉

[1] South China University of Technology, Guangzhou, China
[2] Pengcheng Laboratory, Shenzhen, China

Abstract. Transformer-based object trackers are renowned for their strong performance, yet dense token processing often leads to prohibitive computational cost, limiting real-time deployment on edge devices. While recent works explore token pruning to reduce computation, they often stop short of an end-to-end sparse pipeline, as early-layer token scores can be noisy without a motion prior, and many trackers ultimately fall back to dense reshaping to feed the dense prediction head that partially negates the savings. We introduce Motion-aware Sparse Tracker (MaST), a sparse tracking framework that makes sparsity effective from tokens to boxes. First, MaST injects a lightweight motion prior to refine cross-attention-based importance scores, enabling earlier and more stable token reduction in the search region. Second, we introduce a natively sparse prediction head that operates directly on the retained unstructured tokens with a score-first, regress-once design, eliminating dense padding/reshaping and reducing redundant computation. Extensive experiments on multiple benchmarks demonstrate that MaST establishes new state of the art among lightweight trackers, where MaST-tiny attains 63.8 AUC on LaSOT and 80.1 SUC on TrackingNet, surpassing the prior best AsymTrack-S by +1.0 AUC and +2.2 SUC while running at 152 FPS on Jetson Nano, nearly twice as fast as AsymTrack-S at 88 FPS. Code is available at `github.com/TsingWei/MaST`.

Keywords: Motion Prior · Token Sparsification · Lightweight Object Tracking

1 Introduction

Visual object tracking is a fundamental task in computer vision with broad applications in autonomous driving [18], surveillance [46], augmented reality [50], and robotics [38]. Given the target location in an initial frame, the goal is to estimate its trajectory throughout a video sequence — a task made challenging by occlusions, background clutter, illumination changes, and appearance variations.

✉ Corresponding authors: Quan Tang (`bebdong@gmail.com`) and Fagui Liu (`fgliu@scut.edu.cn`).

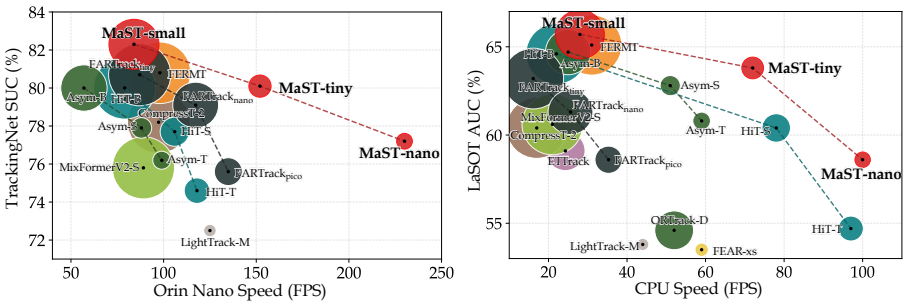

Fig. 1: Speed–accuracy trade-off on edge platforms. The left panel shows TrackingNet SUC vs. Orin Nano speed (FPS). The right panel shows LaSOT AUC vs. CPU speed (FPS). Bubble area is proportional to model parameter count. Our MaST-{small, tiny, nano} achieve Pareto-optimal trade-offs, delivering competitive or superior accuracy at significantly higher speeds compared to prior lightweight trackers.

Recent one-stream Transformer-based trackers [43, 49] have achieved remarkable accuracy by jointly modeling template–search interactions through self-attention. However, their quadratic attention complexity over long token sequences incurs substantial computational cost, hindering deployment on resource-constrained platforms such as UAVs and mobile robots. The field has pursued two lines of work to improve efficiency, including lightweight architectures [5, 24, 48] that employ scaled-down backbones at the expense of discriminative power, and adaptive computation methods [27, 28, 45] that reduce average latency but leave the worst-case computational cost largely unchanged.

Unlike CNNs that operate on dense feature maps, Vision Transformers represent images as discrete token sequences, naturally enabling dynamic token reduction — a strategy proven effective in general vision tasks [4, 29]. This makes token sparsification a particularly promising avenue for accelerating Transformer-based trackers. Several recent works have explored this direction. OSTrack [49] repurposes the cross-attention scores between search and template tokens as a pruning criterion, retaining a fixed ratio of search tokens at intermediate encoder layers; FARTrack [40] extends this by additionally leveraging attention to learnable location tokens; AVTrack [28] trains an auxiliary sub-network to predict per-token importance for early dropping decisions.

Despite these advances, existing token sparsification approaches for tracking still face systematic limitations. On the token-reduction side, pruning is typically applied too late, as illustrated in Fig. 2(a), early-layer attention maps are highly diffuse, making attention-based importance scores unreliable and forcing most methods to defer pruning to intermediate layers (e.g., progressively at the 4rd, 7th, 11th), leaving the computationally expensive early layers to process the full token set. Moreover, common pruning criteria (cross-attention scores or auxiliary predictors) capture only appearance-level relevance within the current frame and ignore the strong temporal motion continuity in tracking, i.e., the prior

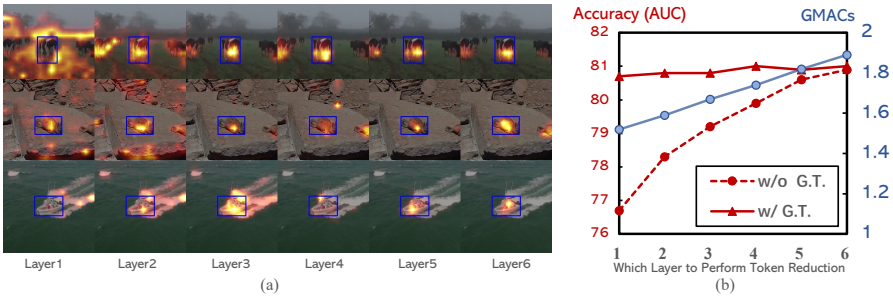

Fig. 2: Why early-stage token pruning fails. (a) Attention map visualization across encoder layers on three tracking sequences. Early layers produce diffuse, noisy attention that has not yet localized the target, explaining why existing methods conservatively defer token pruning to intermediate layers. (b) Effect of performing one-shot token reduction (33% retention align to OSTrack [49]) at different encoder layers on GOT-10k *val*. Without spatial guidance (*w/o G.T.*), accuracy drops sharply as pruning moves to earlier layers. With ground-truth target locations guiding selection (*w/ G.T.*), accuracy remains nearly flat across all layers while GMACs decrease substantially with earlier pruning.

knowledge of where the target was in the previous frame. On the output side, the prediction head remains a dense bottleneck. Convolutional heads assume a dense, spatially structured token grid, so aggressive sparsification requires padding and reshaping sparse tokens back to a full 2D feature map, wasting computation on empty positions; at high compression ratios, the target-center token may even be pruned, leading to severe mislocalization. This mismatch between a sparse backbone and a dense head caps the achievable speedup and degrades accuracy. Notably, the decoding head is not necessarily convolutional in modern trackers, and some methods decode targets via additional learnable tokens (e.g., MixFormerV2 [11] and the ARTrack series [40, 43]), and LoRAT [30] replaces the convolutional head with MLPs. However, these predictor designs are typically developed under dense token/feature assumptions, and their behavior under aggressive token sparsification remains under-explored.

A natural question arises as to whether early-layer pruning is inherently harmful, or whether it merely lacks the right guidance. To answer this, we conduct a diagnostic experiment that disentangles the effect of when to prune from what to prune. As shown in Fig. 2(b), when token selection at 33% retention is guided by ground-truth target locations, accuracy remains remarkably stable regardless of the pruning layer, while computational cost (GMACs) decrease substantially with earlier pruning. In contrast, without such guidance, accuracy degrades sharply as pruning moves to earlier layers. This result indicates that the performance degradation associated with early pruning stems not from insufficient token representations, but from the inability to identify the correct tokens to retain at that stage. The challenge therefore reduces to two sub-problems, (i) obtaining a reliable motion prior for token selection before the

encoder has produced discriminative features, and (ii) designing a prediction head that natively operates on the resulting sparse, unstructured token set.

In this work, we propose **MaST** (**M**otion-**a**ware **S**parse **T**racker), a sparse tracking framework that addresses these limitations with a cohesive design. As shown in Fig. 1, MaST variants achieve Pareto-optimal speed–accuracy trade-offs on edge platforms. MaST performs early, task-driven token reduction in the search region by injecting a history-based motion prior into the selection process. To avoid a dense output bottleneck, MaST also introduces a natively **sparse prediction head** that operates directly on unstructured tokens with a score-first, regress-once design, eliminating dense reshape and reducing redundant computation.

Our main contributions are summarized as follows.

- We propose MaST, a sparse tracking framework that performs early, task-driven token reduction in the search region using temporal motion priors.
- We introduce a natively sparse prediction head with a score-first, regress-once design that operates directly on unstructured tokens, removing the dense head bottleneck.
- Extensive experiments on LaSOT, TrackingNet, GOT-10k, and more benchmarks demonstrate that MaST achieves competitive or superior accuracy while delivering substantially higher measured inference speeds on edge platforms such as Raspberry Pi and Jetson Nano.

2 Related Works

Lightweight Visual Tracking. Modern one-stream Transformer trackers [6, 10, 11, 43, 49] unify feature extraction and relation modeling into a single ViT backbone, achieving strong accuracy but suffering from quadratic self-attention complexity that hinders deployment on edge devices. Prior efforts to close this gap fall into three categories, each with notable limitations. Compact model design [3, 5, 8, 24, 28, 42] reduces parameter count but sacrifices representational capacity. Model compression via layer pruning [11, 42] or knowledge distillation [11, 21] risks non-trivial accuracy degradation. Conditional dynamic computation [27, 28, 45] improves *average* throughput but yields variable per-frame latency, complicating hardware scheduling. In contrast, our approach uses fixed top-K token selection guided by a motion prior, achieving both predictable latency and high accuracy.

Temporal Priors in Visual Tracking. Temporal priors, especially motion locality, are well-established in visual tracking, as target motion between consecutive frames is typically small relative to the search region. Classical approaches exploit this prior via a penalty window (Hanning/cosine) applied to the output score map. SiamFC [1] first applied a cosine window to the cross-correlation response to suppress high-response regions near the boundary of the search area. SiamRPN++ [26] further combines a cosine window penalty with a size-change penalty on the regression response to reduce large, abrupt displacement predictions. Discriminative correlation filters such as KCF [20] and ECO [13] apply a cosine window to the input features to reduce periodic boundary artefacts,

which also implicitly penalises predictions near the edges. Many modern trackers, including DiMP [2], prDiMP [14], STARK [47], and OSTrack [49], likewise multiply a Hanning window onto the predicted score map as a post-processing step before selecting the final target location. Beyond hand-crafted windows, recent Transformer-based trackers also explore data-driven temporal modeling by encoding historical predictions as tokens and letting the network learn to use this information, e.g., SwinTrack [31] and the ARTrack series [40, 43]. However, these designs do not explicitly impose a spatial locality prior during computation. Despite their widespread adoption, penalty windows are applied only after the full forward pass and merely re-weight already-computed scores, leaving the heavy encoder computation unaffected. In contrast, our approach injects temporal motion priors into the token reduction process itself, using them to guide which tokens are computed in subsequent layers and thereby reducing computation.

Token Sparsification in Vision Transformers. The tokenized processing paradigm of Vision Transformers enables novel sparsification strategies that are absent in CNN-based trackers. EViT [29] pioneered token pruning by selecting tokens with the highest attention to the class token. OSTrack [49] adapted this idea by measuring cross-attention between search and template tokens, retaining the top-K most correlated tokens. However, their reliance on early-layer attention scores introduces noise. As a result, many methods apply token pruning conservatively and progressively at intermediate layers, leading to limited speedup. DynamicViT [37] proposed learnable importance predictors using lightweight MLPs, and DToP [39] further explored this idea in semantic segmentation. Similarly, GRM [19] applied such techniques to guide token interactions in tracking tasks. Although these methods improve robustness. AVTrack [28] also uses sub-network for early token exit, but the resulting variable computation per frame complicates hardware scheduling and deployment. In contrast, our approach combines motion-guided locality priors with fixed token budgets (top-K selection), ensuring predictable and consistent latency.

3 Method

3.1 Overview

The proposed framework shown in Fig. 3 operates in three stages, namely motion prediction, token sparsification, and target decoding. Given an input video frame and the target's initial location, a pair of images is processed as input, consisting of a template image denoted as $Z \in \mathbb{R}^{3\times H_z\times W_z}$ and a search image denoted as $X \in \mathbb{R}^{3\times H_x\times W_x}$. These images are divided into non-overlapping patches of size $P \times P$, resulting in $P_z = \frac{H_z\times W_z}{P^2}$ patches for the template and $P_x = \frac{H_x\times W_x}{P^2}$ patches for the search region. After patch embedding concatenated together, features are simultaneously extracted and fusion in the form of sequence using a Transformer encoder, in which the sequence will be first handled by our proposed sparsification block. In the end, the retained tokens will be further processed by the prediction head.

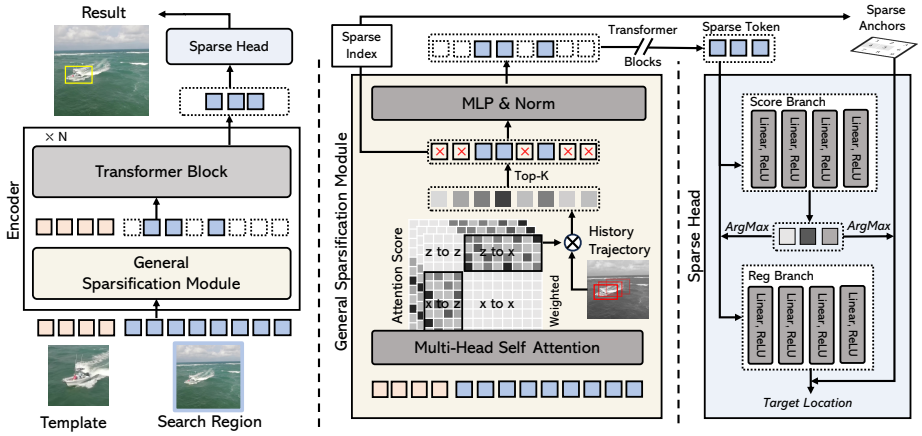

Fig. 3: Overview of the MaST framework. The General Sparsification Module is inserted *once* at an early encoder layer, retaining the top-K search tokens; all subsequent Transformer blocks operate on this reduced set. Token selection fuses cross-attention relevance with a motion prior from the previous prediction for early and stable pruning. The retained sparse tokens are then decoded by a lightweight MLP sparse head with a score-first, regress-once pipeline, eliminating dense reshaping and redundant computation.

3.2 Motion-Aware Token Sparsification

Token sparsification plays a critical role in reducing the computational cost of Transformer-based visual object trackers by selectively focusing on the most relevant tokens. Typically, an importance score is assigned to each search token to determine whether it should be retained. Existing approaches often derive these scores either from an auxiliary prediction network or from cross-attention maps that measure the interaction between search and template tokens. Following OSTrack [49], we adopt the cross-attention map as the basis for token scoring. However, early-layer attention is often diffuse and noisy, which makes naive early pruning unreliable. To enable earlier and more stable token reduction, we inject a lightweight temporal motion prior derived from the previous prediction.

Importance Score. In our framework, the importance score of search tokens is derived from the cross-attention map in the Transformer encoder. Let the search tokens and template tokens be represented as $\mathbf{Q}_x \in \mathbb{R}^{P_x \times d}$ (queries) and $\mathbf{K}_z \in \mathbb{R}^{P_z \times d}$ (keys), where P_x and P_z are the numbers of tokens in the search and template regions, respectively, and d is the feature dimension. Following the approach explored in OSTrack, we simplify the aggregation process by only using the token from the center patch of the template as a representative feature. This reduces unnecessary complexity while retaining strong performance.

The cross-attention map $\mathbf{A}_{x\to z} \in \mathbb{R}^{P_x \times P_z}$, which measures the interaction between search and template tokens, is defined as:

$$\mathbf{A}_{x\to z} = \text{Softmax}\left(\frac{\mathbf{Q}_x \mathbf{K}_z^\top}{\sqrt{d}}\right), \tag{1}$$

where $\mathbf{A}_{x\to z}[i,j]$ represents the attention weight between the i-th search token and the j-th template token. Given that we only use the center token of the template, denoted as $\mathbf{k}_c \in \mathbb{R}^d$, the importance score $\mathbf{s}_i$ for the i-th search token is simplified as:

$$\mathbf{s}_i = \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_c/\sqrt{d}\right)}{\sum_{k=1}^{P_x} \exp\left(\mathbf{q}_k^\top \mathbf{k}_c/\sqrt{d}\right)}, \tag{2}$$

where $\mathbf{q}_i \in \mathbb{R}^d$ is the query feature of the i-th search token. This formulation focuses the importance scoring process on the interaction between search tokens and the central template token, providing a lightweight yet effective means of measuring token relevance.

Motion Prior Injection. Directly using early-layer cross-attention scores for pruning is sub-optimal due to their diffuse and noisy nature. Tracking, however, provides a strong temporal cue, as the target location changes smoothly across frames. We therefore build a simple spatial prior from the previous prediction and use it to guide token selection.

Specifically, as illustrated in the center panel of Fig. 3, we construct a Motion Window over the search region, centered at the previous predicted target location, and use it to softly re-weight the cross-attention importance scores. In practice, we find that a 2D Gaussian provides an effective and lightweight realization. Given the predicted bounding box from the previous frame $\mathbf{b}_{t-1} = (x_{t-1}, y_{t-1}, w_{t-1}, h_{t-1})$, where (x_{t-1}, y_{t-1}) is the box center and w_{t-1}, h_{t-1} are its width and height, the Gaussian is defined as:

$$\mathcal{G}_t(u,v) = \exp\left(-\frac{(u-x_{t-1})^2}{2\sigma_x^2} - \frac{(v-y_{t-1})^2}{2\sigma_y^2}\right), \tag{3}$$

where (u,v) represents the spatial coordinates of a token in the search region, and $\sigma_x = \gamma w_{t-1}$, $\sigma_y = \gamma h_{t-1}$ are the standard deviations of the Gaussian in the x and y directions, respectively. The scaling factor γ controls the size of the window, balancing the trade-off between including potentially relevant tokens and excluding irrelevant ones. We found that setting $\gamma = 0.5$ yields reasonable performance.

The Motion Window assigns higher weights to tokens closer to the previous predicted target center. We refine the importance scores by combining the cross-attention importance scores $\mathbf{s}_i$ with the Motion Window values $\mathcal{G}_t(u_i, v_i)$ for each token:

$$\mathbf{w}_i = \mathcal{G}_t(u_i, v_i) \cdot \mathbf{s}_i, \tag{4}$$

where (u_i, v_i) is the spatial location of the i-th token. This combined score $\mathbf{w}_i$ ensures that tokens within the likely target region (as constrained by the Motion Window) are prioritized, even if their raw attention scores are diffuse. Then we only keep the top-K tokens $\mathcal{L}_K \in \mathbb{R}^{N_K \times d}$ as the output of Motion-Aware Token Sparsification block, attached by the standard Transformer blocks in the encoder.

3.3 Fully Sparse Prediction Head

Our goal is to avoid the dense-head bottleneck when the backbone operates on a sparsified, unstructured token set. Let the retained search tokens be $\mathcal{L}_K = \{\mathbf{f}_k\}_{k=1}^{N_K}$, where $N_K \ll H_x W_x / P^2$ and each token keeps its original 2D location on the patch grid, denoted as $\mathbf{p}_k = (u_k, v_k)$. As illustrated in Fig. 4, unlike dense heads that must pad and reshape sparse tokens back to a full 2D grid before stacked convolutions, we design a natively sparse head that directly consumes $\mathcal{L}_K$ without padding or reshaping back to a dense feature map. We parameterize both branches with lightweight MLPs, similar to prior MLP-based heads (e.g., LoRAT [30]). We further tailor the head for sparse token decoding via a score-first, regress-once computation path.

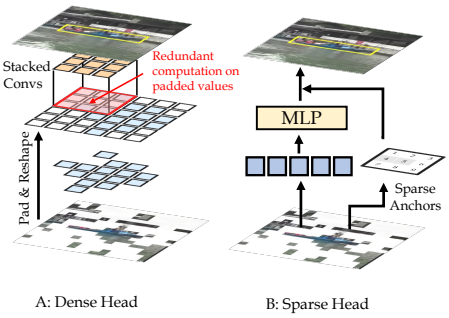

Fig. 4: Dense vs. sparse prediction head. (A) Dense head pads and reshapes sparse tokens into a full 2D grid, wasting computation on empty positions. (B) Our sparse head applies an MLP directly on retained tokens via sparse anchors, with no padding or reshape.

Score-first, regress-once. As illustrated in Fig. 3, the head contains two lightweight branches.

- **Score branch** g_{s} predicts a scalar confidence $s_k \in \mathbb{R}$ for each retained token. The target token is selected by

$$k^* = \arg\max_k \; s_k. \tag{5}$$

  This selection can be implemented by scattering $\{s_k\}$ to a sparse score map aligned with the original grid and applying an `ArgMax`, but no dense feature computation is required.
- **Regression branch** g_{r} predicts box parameters only *once* for the selected token $\mathbf{f}_{k^*}$. Denoting the regression output as Δ_{k^*}, the final box is decoded as

$$\hat{\mathbf{b}} = \mathrm{Decode}(\mathbf{p}_{k^*}, \Delta_{k^*}). \tag{6}$$

Importantly, the anchor point $\mathbf{p}_{k^*}$ is directly retrieved from the retained token's stored coordinate (i.e., a sparse map lookup), avoiding any padding/reshape to dense 2D feature maps; meanwhile, no regression is computed for other tokens.

This design makes the head complexity scale with N_K for scoring and constant-time for regression, matching the sparsified backbone and eliminating redundant computation on empty spatial positions.

Training objective. We supervise the score branch with a classification loss over sparse tokens, and supervise the regression branch on a single token aligned with the ground-truth target. Let $\mathbf{c}$ be the ground-truth target center on the patch grid and define

$$k^{\mathrm{gt}} = \arg\min_k \|\mathbf{p}_k - \mathbf{c}\|_2. \tag{7}$$

The overall head loss is

$$L_{\mathrm{head}} = L_{\mathrm{cls}}(\{s_k\}) + \lambda_{\ell_1} L_{\ell_1}(\hat{\mathbf{b}}_{k^{\mathrm{gt}}}, \mathbf{b}) + \lambda_{\mathrm{GIoU}} L_{\mathrm{GIoU}}(\hat{\mathbf{b}}_{k^{\mathrm{gt}}}, \mathbf{b}), \tag{8}$$

where $\mathbf{b}$ is the ground-truth box, and we set $\lambda_{\mathrm{GIoU}} = 2$ and $\lambda_{\ell_1} = 5$.

4 Experiment

Device. Our MaST models are implemented using Python 3.10 and PyTorch 2.4.1. Training is conducted on a single NVIDIA RTX 3090 GPU. We evaluate inference speed on three representative edge platforms, namely Raspberry Pi 5, Apple M4, and NVIDIA Jetson Orin Nano. To ensure a consistent evaluation environment and better reflect real-world application deployment, we export ONNX files with OpSet 17 and benchmark with ONNXRuntime 1.20.1 to report practical on-device performance. For trackers whose source code is publicly available, we re-evaluate their speed under the same protocol to ensure a fair comparison.

Table 1: Details of our MaST model variants.

Model	MaST		
	nano	tiny	small
Backbone	ViT-Tiny		
Encoder Layers	8	12	12
Template Size	128	128	192
Search Size	256	256	384
Token Rate (%)	30	30	30
MACs (G)	0.59	0.84	1.82
Params (M)	3.85	5.63	5.63

Model. To demonstrate our method on the low-cost edge device, we use ViT-Tiny [16] as the encoder and the training are initialized with the MAE-lite [41]. We provide three model variants with different backbones, encoder depths, and token retention rates, as summarized in Tab. 1. MaST-nano uses the first 8 layers of ViT-Tiny with a 20% retention rate for maximum efficiency; MaST-tiny employs the full 12-layer ViT-Tiny with 30% retention; MaST-small scales to ViT-Small with 30% retention for higher accuracy.

Training. The training splits of COCO [32], LaSOT [17], GOT-10k [22] TrackingNet [35] and VastTrack [36] are used for training. Common data augmentations including horizontal flip and brightness jittering are used in training. The

Table 2: State-of-the-art comparisons of efficient tracking on three large-scale benchmarks, grouped by computational cost. Red and blue denote the best and second-best results within each group. Gray FPS values indicate speeds below VOT realtime setting (20 FPS).

Method	Pub.	LaSOT			TrackingNet			GOT-10k			MACs	FPS		
		AUC	P_{Norm}	P	SUC	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$	(G)	RPi5	CPU	Nano
KCF [20]	TPAMI'14	21.1	22.8	18.4	-	-	-	27.9	26.3	9.9	-	142	946	203[3]
∼500M MACs														
MaST-nano	Ours	58.6	67.7	59.4	77.2	82.5	72.4	61.6	71.8	53.7	0.585	30.1	101	230
AsymTrack-T [52]	AAAI'25	60.8	68.7	61.2	76.2	80.9	71.6	62.3	71.3	54.7	0.708	18.1	60	99
FEAR-XS [5]	ECCV'22	53.5	-	54.5	-	-	-	61.9	72.2	-	0.532	19.5	59	146
LightTrack [48]	CVPR'21	53.8	-	53.7	72.5	77.8	69.5	61.1	71.0	-	0.483	13.4	44	124
∼1G MACs														
MaST-tiny	Ours	63.8	72.2	67.2	80.1	85.3	77.2	66.6	76.0	62.5	0.836	22.6	72	152
$FARTrack_{pico}$ [40]	ICLR'26	58.6	67.1	59.6	75.6	81.3	70.5	62.8	72.6	50.9	1.08	17.2	35	134
AsymTrack-S [52]	AAAI'25	62.8	71.2	64.8	77.9	82.2	74.0	65.5	74.8	58.9	0.806	15.6	51	88
ORTrack-D [44]	CVPR'25	54.6	62.6	54.3	73.7	79.1	68.2	-	-	-	1.54	18.5	52	138
HiT-Small [24]	ICCV'23	60.5	68.3	61.5	77.7	81.9	73.1	62.6	71.2	54.4	1.13	21.5	78	106
HiT-Tiny [24]	ICCV'23	54.8	60.5	52.9	74.6	78.1	68.8	52.6	59.3	42.7	0.995	27.3	97	118
E.T.Track [3]	WACV'23	59.1	-	-	75.0	80.3	70.6	-	-	-	1.56	8.8	24	71
∼2G MACs														
MaST-small	Ours	65.8	74.7	70.4	82.3	87.1	80.5	70.0	80.4	67.0	1.82	7.5	30	98
$FARTrack_{nano}$ [40]	ICLR'26	61.3	69.7	64.1	79.1	84.5	75.6	69.9	81.2	61.4	1.78	10.4	25	117
$FARTrack_{tiny}$ [40]	ICLR'26	63.2	71.6	66.7	80.7	85.6	77.5	70.6	81.0	63.8	2.65	6.9	16	87
AsymTrack-B [52]	AAAI'25	64.7	73.0	67.8	80.0	84.5	77.4	67.7	76.6	61.4	1.81	6.2	25	57
FERMT [51]	ECCV'24	65.1	74.6	69.1	80.8	80.9	78.1	69.6	80.1	63.2	2.31	7.9	31	84
HiT-Base [24]	ICCV'23	64.6	73.3	68.1	80.0	84.4	77.3	64.0	72.1	58.1	4.34	6.7	22	79
Heavy Trackers														
SUTrack-T [7]	AAAI'25	69.6	79.3	75.4	82.7	87.2	80.8	72.7	82.1	70.5	5.52	2.1	10.9	23
MCITrack-T [23]	AAAI'25	71.7	81.5	78.2	84.8	89.4	83.7	74.0	83.9	72.1	12.6	-	4.3	9.1
LoRAT [30]	ECCV'24	71.7	80.9	77.3	83.5	87.9	82.1	72.1	81.8	70.7	19.1	-	5.4	12.1
OSTrack [49]	ECCV'22	69.1	78.7	75.2	83.1	87.8	82.0	71.0	80.4	68.2	18.6	-	6.5	13.7

GPU holds 128 image pairs as the batch size. We train the model with AdamW optimizer [33], set the weight decay to 10^{-4}, the initial learning rate for the backbone to 4×10^{-5} and other parameters to 4×10^{-4}, respectively. The total training epochs are set to 300 with 60k image pairs per epoch and we decrease the learning rate by a factor of 10 after 240 epochs. The sparsification rate is gradually warmed-up during training following OSTrack [49]. More details of training and hyper-parameters can be found in the supplementary material.

4.1 Comparison with State-of-the-arts

We conduct a comprehensive comparison on five widely used tracking benchmarks. Trackers are grouped into three tiers, namely ∼500M, ∼1G, and ∼2G MACs, based on their computational cost, and compared within the same tier.

[3] KCF is tested on the CPU of Jetson Orin Nano with 256 input size.

Table 3: Comparisons with lightweight trackers on more benchmarks.

Method	FPS		$\text{LaSOT}_{\text{ext}}$	UAV123	NFS
	RPi	Nano	AUC	AUC	AUC
LightTrack [48]	13.4	124	-	62.5	55.3
FEAR-XS [5]	19.5	146	-	61.4	61.4
E.T.Track [3]	8.8	71	-	59.0	59.0
HiT-Base [24]	6.7	79	44.1	65.6	63.6
MixformerV2-S [11]	7.1	89	43.6	65.8	-
LiteTrack-B4 [42]	3.6	45	-	66.4	63.4
CompressTracker-2 [21]	6.1	97	40.4	62.5	-
AsymTrack-B [52]	6.2	57	44.6	66.5	64.4
$\text{FARTrack}_{\text{tiny}}$ [40]	6.9	87	45.0	65.8	66.9
MaST-nano	30.1	230	41.1	62.9	64.4
MaST-tiny	22.6	152	42.8	66.6	66.2
MaST-small	7.5	98	45.3	66.4	65.7

Table 4: Comparison on the VastTrack [36] benchmark.

Method	VastTrack		
	SUC	P_{Norm}	Prec
ATOM [12]	17.2	14.1	10.2
SiamRPN++ [26]	28.1	29.7	24.9
TransT [9]	29.9	31.4	25.4
STARK [47]	33.4	34.3	30.8
OSTrack [49]	33.6	34.5	31.5
ARTrack	35.6	35.8	32.4
MixFormerV2-B [11]	35.2	36.5	33.0
$\text{FARTrack}_{\text{pico}}$ [40]	30.3	33.0	25.7
$\text{FARTrack}_{\text{nano}}$ [40]	33.9	35.1	30.3
$\text{FARTrack}_{\text{tiny}}$ [40]	35.2	36.5	32.3
MaST-nano	30.6	37.8	30.1
MaST-tiny	33.4	40.7	26.2
MaST-small	35.6	43.2	33.0

LaSOT. LaSOT [17] is a large-scale long-term tracking benchmark comprising 1,400 video sequences with 280 sequences reserved for testing. As shown in Tab. 2, MaST-tiny achieves the highest AUC of 63.8 and P_{Norm} of 72.2 in the ∼1G group, outperforming AsymTrack-S [52] by +1.0 AUC and HiT-Small [24] by +3.3 AUC, while running at 22.6 FPS on Raspberry Pi 5. MaST-small further pushes to 65.8 AUC in the ∼2G group, surpassing FERMT [51] by +0.7 AUC.

TrackingNet. TrackingNet [35] is a large-scale short-term benchmark with 511 test sequences covering diverse real-world scenarios. MaST-tiny leads the ∼1G group with 80.1 SUC and P_{Norm} of 85.3, surpassing AsymTrack-S by +2.2 SUC. In the ∼2G group, MaST-small achieves 82.3 SUC, outperforming all competing methods in its tier.

GOT-10k. GOT-10k [22] is a large-scale dataset with 180 test sequences featuring non-overlapping object categories between train and test splits, designed to measure generalization. MaST-tiny attains an AO of 66.6 and $\text{SR}_{0.75}$ of 62.5 in the ∼1G group, outperforming AsymTrack-S by +1.1 AO and +3.6 $\text{SR}_{0.75}$. MaST-small achieves 70.0 AO and the highest $\text{SR}_{0.75}$ of 67.0 in the ∼2G group.

$\text{LaSOT}_{\text{ext}}$. $\text{LaSOT}_{\text{ext}}$ [17] extends LaSOT with 150 test sequences covering more diverse and challenging long-term scenarios. As reported in Tab. 3, MaST-small achieves the highest AUC of 45.3, surpassing $\text{FARTrack}_{\text{tiny}}$ [40] by +0.3 and AsymTrack-B [52] by +0.7. MaST-tiny achieves 42.8 AUC at 22.6 FPS on Raspberry Pi 5, over 3× faster than AsymTrack-B (6.2 FPS) ,offering a favorable speed–accuracy trade-off for long-term tracking.

UAV123. UAV123 [34] is an aerial tracking benchmark with 123 low-altitude UAV sequences featuring fast motion, small targets, and frequent occlusion. MaST-tiny achieves the highest AUC of 66.6 among all compared methods, outperforming LiteTrack-B4 [42] (66.4) and AsymTrack-B [52] (66.5) while running at 22.6 FPS on Raspberry Pi 5. MaST-nano attains 62.9 AUC at a leading 30.1 FPS, making it well-suited for resource-constrained aerial applications.

Table 5: Comparison of performance on various sparisfication on encoder.

Sparsification By	LaSOT			Encoder MACs	FPS		
	AUC	P_{Norm}	P		RPi5	CPU	Nano
None	64.0	74.2	68.3	1752M	9.1	37	94
Attention	60.5	71.9	65.6	824M	22.9	73	157
Prediction	59.4	71.4	64.9	874M	21.5	66	138
Attention + Motion	63.8	73.6	67.9	824M	22.6	72	152
Motion	62.6	72.2	66.4	824M	23.5	78	169

Table 6: Comparison of prediction heads on LaSOT with ViT-Tiny backbone.

Prediction Head	AUC		MACs (G)		RPi FPS	
	Dense	Sparse	Dense	Sparse	Dense	Sparse
Global Localization						
Loc Tokens [11]	59.0	57.9	1.76	0.845	10.0	23.9
Trans. Decoder	61.4	60.1	1.88	0.833	9.4	22.7
Anchor-based Localization						
3×3 Conv [49]	65.0	64.1	2.39	1.463	7.7	13.8
MLP (Dense) [30]	64.0	63.8	1.81	0.872	9.1	21.3
MLP (Sparse)	64.0	63.8	1.75	0.836	9.7	23.2

NFS. NFS [25] is a high-frame-rate benchmark with 100 sequences featuring fast-moving objects that challenge tracker responsiveness. Under 30fps setting, MaST-tiny achieves 66.2 AUC, second only to FARTrack$_{\text{tiny}}$ [40] (66.9) while running substantially faster on edge devices. MaST-small achieves 65.7 AUC, surpassing AsymTrack-B [52] (64.4) and HiT-Base [24] (63.6).

VastTrack. VastTrack [36] is a large-vocabulary benchmark spanning 2,115 object categories with over 50,000 sequences, designed to evaluate tracking generalization across a vast range of target types. As shown in Tab. 4, MaST-small achieves a SUC of 35.6 and P_{Norm} of 43.2, surpassing heavier MixFormerV2-B [11](35.2 SUC) by +0.4. The P_{Norm} advantage is particularly striking at +6.7 over the next-best method, suggesting superior localization accuracy normalized by target scale.

4.2 Ablation Study

Impact of Different Sparsification Strategies. Tab. 5 compares various token sparsification strategies. Naive pruning based solely on attention scores ("Attention") achieves significant compute reduction but suffers a severe accuracy drop, revealing that early cross-attention scores are too diffuse to reliably identify important tokens. Replacing attention with an MLP predictor ("Prediction") is even worse, as it lacks spatial grounding. The proposed motion prior is the key enabler, as fusing attention scores with the Gaussian spatial prior ("Attention + Motion") nearly closes the accuracy gap to the dense baseline while preserving the full compute savings. Using the motion window alone ("Motion") yields a middle

ground, confirming that the fusion of both signals is optimal. These results underscore that a lightweight temporal prior is critical to making aggressive early-layer sparsification practically viable.

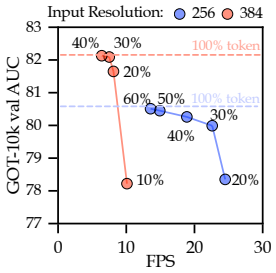

Fig. 5: AUC–FPS trade-off at different token retention rates on GOT-10k val

Analysis of Different Prediction Heads. Tab. 6 compares prediction heads under two paradigms. *Global localization* heads (Loc Tokens, Trans. Decoder) are naturally sparse-compatible and achieve good speedups, but their accuracy is limited (57.9–60.1 AUC) due to insufficient spatial prior. *Anchor-based* heads yield higher accuracy (63.8–64.1 AUC), yet the conventional 3×3 Conv-stacked head must reshape sparse tokens back to a dense grid, incurring redundant computation and capping throughput at 13.8 FPS. Our sparse MLP head operates score-first, regress-once on retained tokens, improving throughput to 23.2 FPS with no accuracy loss over the dense MLP counterpart [30] (63.8 AUC in both cases), confirming the value of a natively sparse head design.

Impact of Token Retention Rate. Fig. 5 examines how the token retention rate affects the AUC–FPS trade-off at two input resolutions on LaSOT. At 384 resolution, retaining only 30–40% of tokens already matches the dense baseline in AUC while substantially boosting throughput; further reduction to 20% causes only a moderate drop. At 256 resolution, 50–60% retention achieves near-lossless accuracy, and 30% retention remains competitive.

In both settings the motion prior guides early token selection reliably regardless of input scale, confirming that MaST's accuracy–efficiency trade-off is robust across resolution choices.

Table 7: Effect of sparsification layer on tracking accuracy and efficiency.

Sparsify at Layer	LaSOT AUC	RPi FPS	MACs (M)
1	63.82	22.6	836
2	63.87	17.6	942
3	63.65	14.0	1004
4	63.83	12.1	1080
5	63.91	11.3	1154
6	63.96	10.1	1238

Layer-wise Sparsification Analysis. Tab. 7 evaluates the effect of applying token sparsification at different layers (all with motion window) of the ViT-Tiny encoder. Under the motion prior, the choice of sparsification layer has only a negligible impact on accuracy: AUC varies by at most 0.4 points across all configurations. In contrast, the speed penalty of delaying sparsification is substantial, throughput nearly halves from 22.6 to 10.1 FPS when sparsification is pushed to later layers. Sparsifying at Layer 1 therefore offers the best cost-effectiveness, as it matches

the accuracy of later-layer alternatives while delivering the highest throughput (22.6 FPS) and the lowest compute (836M FLOPs), and we adopt it as our default.

Comparison with Alternative Compression Strategies. Tab. 8 compares MaST against three compression baselines under an identical compute budget (≈1G MACs, ViT-Tiny backbone): the full dense model (1752M MACs), progressive token pruning from OSTrack [49], resolution reduction from LoReTrack [15], and layer pruning from CompressTracker [21]. Progressive token pruning retains accuracy (64.0 AUC) but provides negligible speedup (9.1→10.5 FPS on RPi5) due to its multi-stage, non-one-shot design. Resolution reduction and layer pruning achieve comparable FPS (∼24 FPS on RPi5) but at a severe accuracy cost of −5.5 and −4.9 AUC, respectively. Our one-shot token sparsification achieves the lowest MACs (836M) and the best AUC (63.8), within 0.2 of the dense baseline, while maintaining competitive speed (22.6 FPS on RPi5). This demonstrates that spatially-guided one-shot token sparsification offers a clearly superior accuracy–efficiency trade-off compared to resolution reduction or model depth compression.

Table 8: Comparison with alternative model compression strategies under the same computational budget (≈1G MACs, ViT-Tiny backbone). "↓ Resolution" reduces the input image size to (48,96); "↓ Layer" compresses the encoder into 4 layers; our method applies motion-aware token sparsification at Layer 1 with 30% retention.

Compression Method	From	MACs (M)	LaSOT AUC	LaSOT P_{Norm}	LaSOT P	FPS RPi5	FPS CPU	FPS Nano
Full model	-	1752	64.0	74.2	68.3	9.1	37	94
↓ Token (Progressive)	OSTrack [49]	1293	64.0	74.1	68.1	10.5	41	106
↓ Resolution	LoReTrack [15]	990	58.5	67.7	59.4	23.6	83	176
↓ Layer	CompressTracker [21]	937	59.1	68.5	60.1	24.1	89	177
↓ Token (One-shot)	Ours	836	63.8	73.6	67.9	22.6	76	152

Visualization of the token sparsification process. As shown in Fig. 6, without a motion prior, raw attention scores are diffuse at early layers, causing retained tokens to scatter across irrelevant background regions. Injecting the motion prior re-ranks scores toward the predicted target location, yielding compact, target-centered retention that directly reduces wasted computation on empty positions.

5 Conclusion

In this work, we present MaST, a sparse tracker that unifies motion-aware early token pruning with a natively sparse prediction head. Unlike prior methods that defer pruning to intermediate layers, a motion prior enables reliable one-shot reduction at the first encoder layer, letting all subsequent Transformer blocks operate on the compressed token set. A score-first, regress-once head decodes

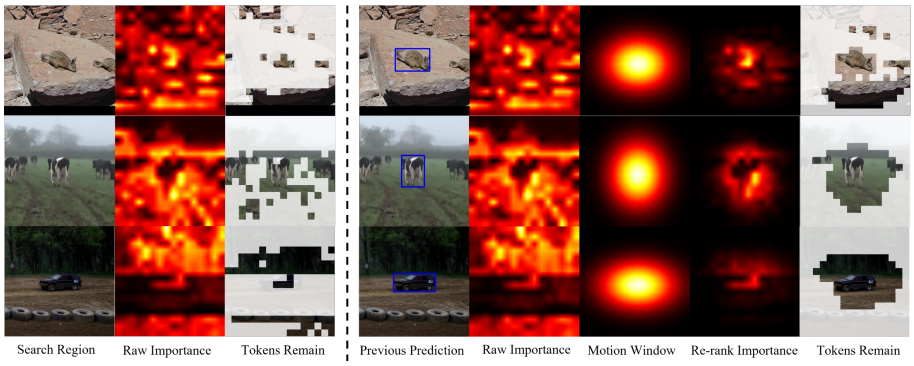

Fig. 6: Token sparsification visualization. *Left* (baseline) shows Search Region, Raw Importance, and Tokens Retained. *Right* (ours) shows Previous Prediction, Raw Importance, Motion Window, Re-ranked Importance, and Tokens Retained. Three tracking sequences are shown per row.

targets directly from retained tokens, eliminating dense reshape and redundant regression. Together, these designs achieve Pareto-optimal speed–accuracy trade-offs on edge platforms, showing that end-to-end sparsity is a practical path to real-time tracking. A remaining limitation is that high-resolution inputs still incur substantial attention computation before sparsification takes effect, motivating input-adaptive pruning as future work.

Acknowledgements

This research is supported by the Major Key Project of Pengcheng Laboratory (PCL2025A13, PCL2025A08), the National Natural Science Foundation of China (U24B20151), and the Guangdong Major Project of Basic and Applied Basic Research (2019B030302002).

References

1. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European conference on computer vision. pp. 850–865. Springer (2016)
2. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6182–6191 (2019)
3. Blatter, P., Kanakis, M., Danelljan, M., Van Gool, L.: Efficient Visual Tracking with Exemplar Transformers. In: WACV. pp. 1571–1581 (2023)
4. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your ViT but faster. In: International Conference on Learning Representations (2023)

5. Borsuk, V., Vei, R., Kupyn, O., Martyniuk, T., Krashenyi, I., Matas, J.: FEAR: Fast, Efficient, Accurate and Robust Visual Tracker. In: ECCV. pp. 644–663 (2022)
6. Chen, B., Li, P., Bai, L., Qiao, L., Shen, Q., Li, B., Gan, W., Wu, W., Ouyang, W.: Backbone is all your need: A simplified architecture for visual object tracking. In: European Conference on Computer Vision. pp. 375–392. Springer (2022)
7. Chen, X., Kang, B., Geng, W., Zhu, J., Liu, Y., Wang, D., Lu, H.: Sutrack: Towards simple and unified single object tracking. In: AAAI (2025)
8. Chen, X., Kang, B., Wang, D., Li, D., Lu, H.: Efficient Visual Tracking via Hierarchical Cross-Attention Transformer. In: ECCVW. pp. 461–477 (2022)
9. Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H.: Transformer tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8126–8135 (2021)
10. Cui, Y., Jiang, C., Wang, L., Wu, G.: 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)
11. Cui, Y., Song, T., Wu, G., Wang, L.: Mixformerv2: Efficient fully transformer tracking. In: NeurIPS. pp. 58736–58751 (2023)
12. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4660–4669 (2019)
13. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Eco: Efficient convolution operators for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6638–6646 (2017)
14. Danelljan, M., Gool, L.V., Timofte, R.: Probabilistic regression for visual tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7183–7192 (2020)
15. Dong, S., Feng, Y., Liang, J., Yang, Q., Lin, Y., Fan, H.: Efficient and accurate low-resolution transformer tracking. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 20677–20684. IEEE (2025)
16. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
17. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
18. Gao, M., Jin, L., Jiang, Y., Guo, B.: Manifold siamese network: A novel visual tracking convnet for autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems **21**(4), 1612–1623 (2020)
19. Gao, S., Zhou, C., Zhang, J.: Generalized relation modeling for transformer tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18686–18695 (2023)
20. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis and machine intelligence **37**(3), 583–596 (2014)
21. Hong, L., Li, J., Zhou, X., Yan, S., Guo, P., Jiang, K., Chen, Z., Gao, S., Li, R., Sheng, X., et al.: General compression framework for efficient transformer object tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13427–13437 (2025)

22. Huang, L., Zhao, X., Huang, K.: 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)
23. Kang, B., Chen, X., Lai, S., Liu, Y., Liu, Y., Wang, D.: Exploring enhanced contextual information for video-level object tracking. In: Proceedings of the AAAI conference on artificial intelligence. vol. 39, pp. 4194–4202 (2025)
24. Kang, B., Chen, X., Wang, D., Peng, H., Lu, H.: Exploring lightweight hierarchical vision transformers for efficient visual tracking. In: ICCV. pp. 9612–9621 (2023)
25. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: 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)
26. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: 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)
27. Li, S., Yang, Y., Zeng, D., Wang, X.: Adaptive and background-aware vision transformer for real-time uav tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 13989–14000 (October 2023)
28. Li, Y., Liu, M., Wu, Y., Wang, X., Yang, X., Li, S.: Learning adaptive and view-invariant vision transformer for real-time uav tracking. In: Forty-first International Conference on Machine Learning (2024)
29. Liang, Y., Ge, C., Tong, Z., Song, Y., Wang, J., Xie, P.: Not all patches are what you need: Expediting vision transformers via token reorganizations. arXiv preprint arXiv:2202.07800 (2022)
30. Lin, L., Fan, H., Zhang, Z., Wang, Y., Xu, Y., Ling, H.: Tracking meets lora: Faster training, larger model, stronger performance. In: ECCV. pp. 300–318. Springer (2024)
31. Lin, L., Fan, H., Zhang, Z., Xu, Y., Ling, H.: Swintrack: A simple and strong baseline for transformer tracking. NeurIPS **35**, 16743–16754 (2022)
32. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, L.: Microsoft coco: Common objects in context. In: ECCV (September 2014)
33. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR. pp. 1–9 (2018)
34. Mueller, M., Smith, N., Ghanem, B.: A benchmark and simulator for uav tracking. In: ECCV. pp. 445–461. Springer (2016)
35. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 300–317 (2018)
36. Peng, L., Gao, J., Liu, X., Li, W., Dong, S., Zhang, Z., Fan, H., Zhang, L.: Vasttrack: Vast category visual object tracking. Advances in Neural Information Processing Systems **37**, 130797–130818 (2024)
37. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems **34**, 13937–13949 (2021)
38. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Autonomous Robots **40**(4), 729–760 (2016)
39. Tang, Q., Zhang, B., Liu, J., Liu, F., Liu, Y.: Dynamic token pruning in plain vision transformers for semantic segmentation. In: ICCV. pp. 777–786 (2023), https://doi.org/10.1109/ICCV51070.2023.00078
40. Wang, G., Lin, T., Bai, Y., Cao, A., Liang, S., Zhao, W., Wei, X.: Fartrack: Fast autoregressive visual tracking with high performance. ICLR (2026)

41. Wang, S., Gao, J., Li, Z., Zhang, X., Hu, W.: A closer look at self-supervised lightweight vision transformers. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 35624–35641. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/wang23e.html
42. Wei, Q., Zeng, B., Liu, J., He, L., Zeng, G.: Litetrack: Layer pruning with asynchronous feature extraction for lightweight and efficient visual tracking. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). pp. 4968–4975 (2024). https://doi.org/10.1109/ICRA57147.2024.10610022
43. Wei, X., Bai, Y., Zheng, Y., Shi, D., Gong, Y.: Autoregressive visual tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9697–9706 (2023)
44. Wu, Y., Wang, X., Yang, X., Liu, M., Zeng, D., Ye, H., Li, S.: Learning occlusion-robust vision transformers for real-time uav tracking. In: CVPR (2025)
45. Wu, Y., Wang, X., Zeng, D., Ye, H., Xie, X., Zhao, Q., Li, S.: Learning motion blur robust vision transformers with dynamic early exit for real-time uav tracking. arXiv preprint arXiv:2407.05383 (2024)
46. Xing, J., Ai, H., Lao, S.: Multiple human tracking based on multi-view upper-body detection and discriminative learning. In: 2010 20th International Conference on Pattern Recognition. pp. 1698–1701. IEEE (2010)
47. Yan, B., Peng, H., Fu, J., Wang, D., Lu, H.: Learning spatio-temporal transformer for visual tracking. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10448–10457 (2021)
48. Yan, B., Peng, H., Wu, K., Wang, D., Fu, J., Lu, H.: Lighttrack: Finding lightweight neural networks for object tracking via one-shot architecture search. In: CVPR 2021 (June 2021)
49. Ye, B., Chang, H., Ma, B., Shan, S., Chen, X.: Joint feature learning and relation modeling for tracking: A one-stream framework. In: European conference on computer vision. pp. 341–357. Springer (2022)
50. Zhang, G., Vela, P.A.: Good features to track for visual slam. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1373–1382 (2015)
51. Zheng, J., Liang, M., Huang, S., Ning, J.: Exploring the feature extraction and relation modeling for light-weight transformer tracking. In: European Conference on Computer Vision. pp. 110–126. Springer (2025)
52. Zhu, J., Tang, H., Chen, X., Wang, X., Wang, D., Lu, H.: Two-stream beats one-stream: Asymmetric siamese network for efficient visual tracking. In: AAAI. vol. 39, pp. 10959–10967 (2025)