SwinTrack: A Simple and Strong Baseline for Transformer Tracking

Liting Lin[1,2*] **Heng Fan**[3*] **Zhipeng Zhang**[4] **Yong Xu**[1,2] **Haibin Ling**[5]
[1]School of Computer Science & Engineering, South China Univ. of Tech., Guangzhou, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]Department of Computer Science and Engineering, University of North Texas, Denton, USA
[4]DiDi Chuxing, Beijing, China
[5]Department of Computer Science, Stony Brook University, Stony Brook, USA
l.lt@mail.scut.edu.cn, heng.fan@unt.edu, zhipeng.zhang.cv@outlook.com
yxu@scut.edu.cn, hling@cs.stonybrook.edu

Abstract

Recently Transformer has been largely explored in tracking and shown state-of-the-art (SOTA) performance. However, existing efforts mainly focus on fusing and enhancing features generated by convolutional neural networks (CNNs). The potential of Transformer in representation learning remains under-explored. In this paper, we aim to further unleash the power of Transformer by proposing a simple yet efficient fully-attentional tracker, dubbed **SwinTrack**, within classic Siamese framework. In particular, both representation learning and feature fusion in SwinTrack leverage the Transformer architecture, enabling better feature interactions for tracking than pure CNN or hybrid CNN-Transformer frameworks. Besides, to further enhance robustness, we present a novel motion token that embeds historical target trajectory to improve tracking by providing temporal context. Our motion token is lightweight with negligible computation but brings clear gains. In our thorough experiments, SwinTrack exceeds existing approaches on multiple benchmarks. Particularly, on the challenging LaSOT, SwinTrack sets a new record with **0.713** SUC score. It also achieves SOTA results on other benchmarks. We expect SwinTrack to serve as a solid baseline for Transformer tracking and facilitate future research. Our codes and results are released at `https://github.com/LitingLin/SwinTrack`.

1 Introduction

Visual tracking has seen considerable progress since deep learning. In particular, the recent Transformer [38] has significantly pushed the state-of-the-art in tracking owing to its ability in modeling long-range dependencies. However, existing methods usually leverage Transformer for fusing and enhancing features generated from convolutional neural networks (CNNs), *e.g.*, ResNet [18]. The potential of exploiting Transformer for feature representation learning is largely under-explored.

Recently, Vision Transformer (ViT) [9] has exhibited great potential in robust feature representation learning. Particularly, its extension Swin Transformer [28] has achieved state-of-the-art (SOTA) results on multiple tasks. Taking inspiration from this, we argue, besides the feature fusion, the representation learning in tracking can also benefit from Transformer via attention. Thus motivated, we propose to develop a fully attentional tracking framework based on Siamese architecture. Specifically, both the feature representation learning and the feature fusion of template and search region are realized by Transformer. More concretely, we borrow the architecture of the powerful

[*]Equal Contributions.

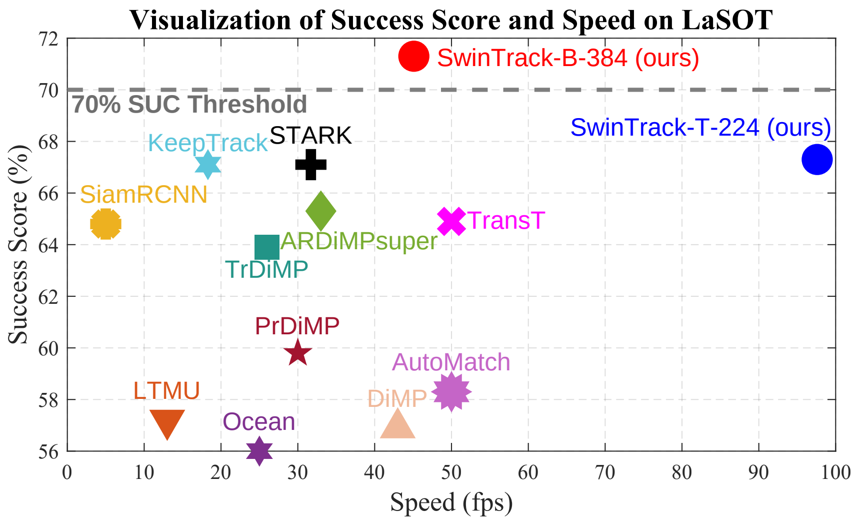


Figure 1: Comparison on LaSOT [11]. Our tracker (SwinTrack-B-384) sets a new record with 0.713 SUC score and still runs efficiently at around 45 *fps*. A lighter version (SwinTrack-T-224) achieves 0.672 SUC score and runs at around 96 *fps*, which is on par with existing SOTAs in accuracy but much faster. Best viewed in color for all figures.

Swin Transformer [28] and adapt it to Siamese tracking. Note that, other Transformer architectures can be used. For feature fusion, we introduce a simple homogeneous concatenation-based fusion architecture, without a query-based decoder.

Moreover, taking into consideration that tracking is a temporal task, we propose a novel motion token to improve robustness. Inspired by that the target usually moves smoothly in a short period, motion token is represented by the historical target trajectory within a local temporal window. We incorporate the (single) motion token in the decoder of feature fusion to leverage motion information during tracking. Despite being conceptually simple, our motion token can effectively boost tracking performance, with negligible computation.

We name our framework SwinTrack. As a pure Transformer framework, SwinTrack enables better interactions inside the feature learning of template and search region and their fusion compared to pure CNN-based [1, 25] and hybrid CNN-Transformer [5, 40, 46] frameworks, leading to more robust performance (see Fig. 1). Fig. 2 demonstrates the architecture of SwinTrack. We conduct extensive experiments on five large-scale benchmarks to verify the effectiveness of SwinTrack, including LaSOT [11], LaSOT$_{\text{ext}}$ [10], TrackingNet [33], GOT-10k [19] and TNL2k [42]. On all benchmarks, SwinTrack achieves promising results and meanwhile runs fast at 45 *fps*. In particular, on the challenging LaSOT, SwinTrack sets a new record of 71.3 SUC score, surpassing the strongest prior tracker [46] (to date) by 3.1 absolute percentage points and crossing the 0.7 SUC threshold *for the first time* (see Fig. 1 again). It also achieves 49.1 SUC, 84.0 SUC, 72.4 AO and 55.9 SUC scores on LaSOT$_{\text{ext}}$, TrackingNet, GOT-10k and TNL2k respectively, which are better than or on par with state-of-the-arts (SoTAs). In addition, we provide a lighter version of SwinTrack that obtains comparable results to SoTAs but runs much faster at around 98 *fps*.

In summary, our contributions are as follows: (**i**) We propose SwinTrack, a simple and strong baseline for fully attentional tracking; (**ii**) We present a simple yet effective motion token, enabling the integration of rich motion context during tracking, further boosting the robustness of SwinTrack, with negligible computation; (**iii**) Our proposed SwinTrack achieves state-of-the-art performance on multiple benchmarks. We believe SwinTrack further shows the potential of Transformer and expect it to serve as a baseline for future research.

2 Related Work

Siamese Tracking. The Siamese tracking methods formulate tracking as a matching problem and aim to offline learn a generic matching function for this task. The seminal method of [1] introduces

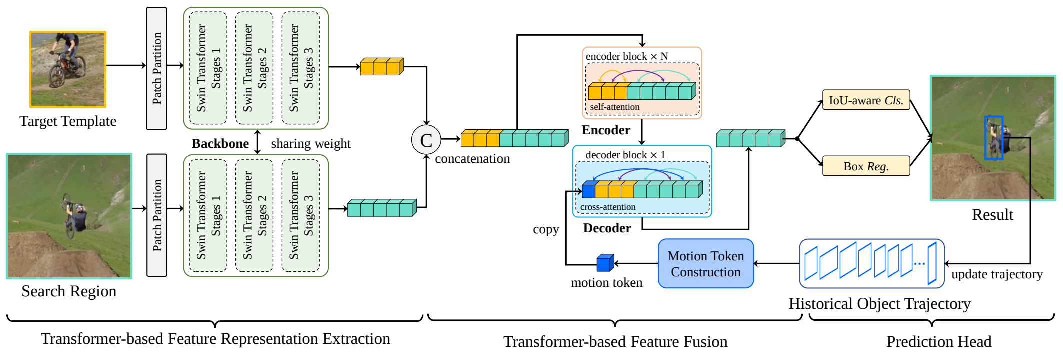


Figure 2: Architecture of SwinTrack, which contains three parts including Transformer-based feature representation extraction, Transformer-based feature fusion and prediction head. Our SwinTrack is a simple and neat tracking framework without complex designs such as multi-scale features or temporal template updating, yet demonstrating state-of-the-art performance. *Best viewed in color*.

a fully convolutional Siamese network for tracking and shows a good balance between accuracy and speed. To improve Siamese tracking in handling scale variation, the work of [25] incorporates the region proposal network (RPN) [34] into the Siamese network and proposes the anchor-based tracker, showing higher accuracy with faster speed. Later, numerous extensions have been presented to improve Siamese tracking, including deeper backbone [24], multi-stage architecture [12, 13], anchor-free Siamese trackers [52], deformable attention [48], to name a few.

Transformer in Vision. Transformer [38] originates from natural language processing (NLP) for machine translation and has been introduced to vision recently and shows great potential. The work of [3] first uses Transformer for object detection and achieved promising results. To explore the capability of Transformer in representation learning, the work of [9] applies Transformer to construct backbone network, and the resulting Vision Transformer (ViT) attains excellent performance compared to convolutional networks while requiring fewer training resources, which encourages many extensions upon ViT[37, 4, 49, 41, 28]. Among them, the Swin Transformer [28] has received extensive attention. It proposes a simple shifted window strategy to replace the fixed-patch method in ViT, which significantly improves efficiency and meanwhile demonstrates state-of-the-art results on multiple image tasks. Our work is inspired by Swin Transformer, but differently, we focus on the video task of visual tracking.

Transformer in Tracking. Inspired by the success in other fields, researchers have leveraged Transformer for tracking. The method of [5] applies Transformer to enhance and fuse features in the Siamese tracking for improvement. The approach of [40] uses Transformer to exploit temporal features to improve tracking robustness. The work of [46] introduces a new transformer architecture dedicated to visual tracking, explores the Spatio-temporal Transformer by integrating the model updating operations into a Transformer module.

Our SwinTrack is related to but significantly different from the above Transformer-based trackers. Specifically, the aforementioned methods mainly apply Transformer to fuse convolutional features and belong to the hybrid CNN-Transformer architecture. Unlike them, SwinTrack is a pure Transformer-based tracking architecture where both representation learning and feature fusion are realized with Transformer, enabling the exploration of better features for robust tracking.

3 Tracking via Vision-Motion Transformer

We present SwinTrack, a vision-motion integrated Transformer for object tracking, in Fig. 2. The proposed framework contains three main components, *i.e.*, the Swin-Transformer backbone for feature extraction, the encoder-decoder network for mixing vision-motion cues, and the head network for localizing targets. In the following sections, we first shortly describe the Swin backbone network, then elaborate on the proposed vision-motion encoder-decoder. Afterward, we give a discussion about our method and shortly describe the network head and training loss.

3.1 Swin-Transformer for Feature Extraction

The deep convolutional neural network has significantly improved the performance of trackers. Along with the advancement of trackers, the backbone network has evolved twice: AlexNet [22] and ResNet [18]. Swin-Transformer [28], in comparison to ResNet, can give a more compact feature representation and richer semantic information to assist succeeding networks in better localizing the target objects (demonstrate in the ablation study demonstrated in the ablation study), which is thus chosen for basic feature extraction in our model.

Our tracker, following Siamese tracking framework [1], requires a pair of image patches as inputs, *i.e.*, template image $\mathbf{z} \in \mathbb{R}^{H_z \times W_z \times 3}$ and search region image $\mathbf{x} \in \mathbb{R}^{H_x \times W_x \times 3}$. As in the typical Swin-Transformer procedure, template and search region images are divided to non-overlapped patches and sent to the network, which generates template tokens (dubbed **T-tokens**) $\varphi(\mathrm{z}) \in \mathbb{R}^{\frac{H_z}{s}\frac{W_z}{s} \times C}$ and search region tokens (dubbed **S-tokens**) $\varphi(\mathrm{x}) \in \mathbb{R}^{\frac{H_x}{s}\frac{W_x}{s} \times C}$. s is the stride of the backbone network. Since there is no dimension projection in our model, C is the hidden dimension of the whole model.

3.2 Vision-Motion Representation Learning

The essential step for matching-based visual tracking is injecting the template information into the search region. In our framework, we adopt an encoder to fuse the features from the template and the search region, meanwhile, a decoder is arranged to achieve vision-motion representation learning, as illustrated in Fig. 2.

Encoder for fusing template and search tokens. The encoder contains a sequence of Transformer blocks where each consists of a multi-head self-attention (MSA) module and a feed-forward network (FFN). FFN contains a two-layers multi-layer perceptron (MLP), GELU activation layer is inserted after the first linear layer. Layer normalization (LN) is always conducted before every module (MSA and FFN). Residual connection is applied to MSA and FFN modules.

Before feeding the features into the encoder, the template and search region tokens are concatenated along spatial dimensions to generate a mixing representation $\mathbf{f}_m$. For each block, the MSA module computes self-attention over mixing union representation, which equals to separately conducting self-attention on T-tokens/S-tokens and meanwhile performing cross-attention between T-tokens and S-tokens, but more efficient. FFN refines the features generated by MSA. When the tokens get out of the encoder, a de-concatenation operation is arranged to decouple the template and search region tokens. The process of encoder can be expressed as:

$$\begin{aligned}
\mathbf{f}_m^1 &= \mathrm{Concat}(\varphi(\mathbf{z}), \varphi(\mathbf{x})) \\
&\cdots \\
\mathbf{f}_m^{l'} &= \mathbf{f}_m^l + \mathrm{MSA}(\mathrm{LN}(\mathbf{f}_m^l)) \\
\mathbf{f}_m^{l+1} &= \mathbf{f}_m^{l'} + \mathrm{FFN}(\mathrm{LN}(\mathbf{f}_m^{l'})) \\
&\cdots \\
\mathbf{f}_z^L, \mathbf{f}_x^L &= \mathrm{DeConcat}(\mathbf{f}^L),
\end{aligned} \tag{1}$$

where l denotes the l-th layer and L denotes the number of blocks.

Decoder for fusing vision and motion information. Before describing the architecture of decoder, we first detail how to generate a motion token (dubbed **M-token**). Motion token is the embedding of the historical trajectory of the target object. The past object trajectory is represented as a set of target object box coordinates, $T = \{\mathbb{o}_1, \mathbb{o}_2, ..., \mathbb{o}_t\}$, where t represents the frame index, $\mathbb{o}$ is the bounding box of target object. $\mathbb{o}$ is defined by the top-left and bottom-right corners of the target object, denotes as $\mathbb{o}_t = (o_t^{x_1}, o_t^{y_1}, o_t^{x_2}, o_t^{y_2})$. For flexible modeling, a sampling process is required to ensure the following properties: **1) fixed length**, **2) focusing on the latest trajectories** and **3) reducing redundancy**. In our method, we sample object trajectory as:

$$\mathcal{T} = \{\mathbb{o}_{s(1)}, \mathbb{o}_{s(2)}, ..., \mathbb{o}_{s(n)}\}, \text{ where } s(i) = max(t - i \times \Delta, 1), \tag{2}$$

n is the number of sampled object trajectories, Δ is the fixed sampling interval. For Siamese tracker, the search region is cropped from the input image. In detail, a cropping with resizing operation can be used to describe the process. Giving the point in the input image as $(\mathbf{x}_i, \mathbf{y}_i)$, the corresponding point in the search region as $(\mathbf{x}_o, \mathbf{y}_o)$, we can formulate the cropping process employed in pre-processing of the Siamese Tracker as $\mathbf{x}_o = (\mathbf{x}_i - i_x)s_x + o_x$ and $\mathbf{y}_o = (\mathbf{y}_i - i_y)s_y + o_y$, where (i_x, i_y) is the center of the cropping window in the input image, (s_x, s_y) is the scaling factor, (o_x, o_y) is the center of cropped and scaled window in the search region. We apply the same transformation on the sampled object trajectory to make the object trajectory invariant to the cropping, denoting $\bar{\mathcal{T}} = \{\bar{\mathbb{o}}_{s(1)}, \bar{\mathbb{o}}_{s(2)}, ..., \bar{\mathbb{o}}_{s(n)}\}$ as the result.

Then, to embed the transformed object trajectory into the network, we adopt four embedding matrices to embed the elements in box coordinates separately. We denotes the embedding matrix as $W \in \mathbb{R}^{(\mathbb{g}+1)\times d}$, $\mathbb{g}$ controls the embedding granularity of the object trajectory, d is the size of each embedding vector. The last entry of the embedding matrix is used as the padding vector, indicating an invalid state, like object absence or out of the search region. Thus, we normalize the sampled target object box coordinates in the range $[1, \mathbb{g}]$, and quantize to integers to get the index of embedding vector:

$$\begin{aligned}
\hat{\mathcal{T}} &= \{\hat{\mathbb{o}}_{s(1)}, \hat{\mathbb{o}}_{s(2)}, ..., \hat{\mathbb{o}}_{s(n)}\}, \\
\text{where } \hat{\mathbb{o}}_{s(i)} &= [\mathbb{n}(\bar{\mathbb{o}}^{x_1}_{s(i)}, w), \mathbb{n}(\bar{\mathbb{o}}^{y_1}_{s(i)}, h), \mathbb{n}(\bar{\mathbb{o}}^{x_2}_{s(i)}, w), \mathbb{n}(\bar{\mathbb{o}}^{y_2}_{s(i)}, h)], \\
\mathbb{n}(o, l) &= \begin{cases} \lfloor \frac{o}{l} \times \mathbb{g} \rfloor & \text{if valid,} \\ \mathbb{g} + 1 & \text{else,} \end{cases}
\end{aligned} \tag{3}$$

where (w, h) is the size of search region feature map.

Finally, the motion token $\mathbf{E}_{motion} \in \mathbb{R}^{1\times d}$ is given by a concatenation of all box coordinate embedding of the sampled object trajectory. FLOPs is negligible because the construction of motion token is just a composition of embedding lookups and token concatenation.

The decoder consists of a multi-head cross-attention(MCA) module and a feed-forward network(FFN). The decoder takes the outputs from the encoder and the motion token as input, generating the final vision-motion representation $\mathbf{f}_{vm} \in \mathbb{R}^{\frac{H_x}{s} \times \frac{W_x}{s} \times C}$ of by computing cross-attention over $\mathbf{f}^L_x$ and $\mathrm{Concat}(\mathbf{E}_{motion}, \mathbf{f}^L_z, \mathbf{f}^L_x)$. The decoder is akin to a layer in the encoder, except that the correlation between the template tokens and the search tokens is dropped since we do not need to update the features from the template image in the last layer. The process of the decoder is formulated as:

$$\begin{aligned}
\mathbf{f}^D_m &= \mathrm{Concat}(\mathbf{E}_{motion}, \mathbf{f}^L_z, \mathbf{f}^L_x) \\
\mathbf{f}'_{vm} &= \mathbf{f}^L_x + \mathrm{MCA}(\mathrm{LN}(\mathbf{f}^L_x), \mathrm{LN}(\mathbf{f}^D_m)) \\
\mathbf{f}_{vm} &= \mathbf{f}'_{vm} + \mathrm{FFN}(\mathrm{LN}(\mathbf{f}'_{vm})).
\end{aligned} \tag{4}$$

$\mathbf{f}_{vm}$ will feed to the head network to generate a classification response map and a bounding box regression map.

Positional encoding. Transformer requires a positional encoding to identify the position of the current processing token[38] because the self-attention module is permutation-invariance. We adopt the *untied positional encoding* [20] as our positional encoding method. The *untied positional encoding* enhances the expressiveness of the model through untie the positional embeddings from token embeddings with an isolated positional embedding matrix. It also considers the case of special tokens, like the motion token in this paper. We generalize the *untied positional encoding* to multi-dimensions multi-sources data to comply with *concatenated-based fusion* in our tracker. See the appendix for the details.

3.3 Discussion

Why concatenated attention? To simplify the description, we call the method described above *concatenation-based fusion*. To fuse and process features from multiple sources, it is intuitive to perform self-attention on the feature from each source separately and then compute cross-attention across features from different sources. We call this method *cross-attention-based fusion*. Transformer makes fewer assumptions about the spatial structure of data, which provides great modeling flexibility.

In comparison to *cross-attention-based fusion*, *concatenation-based fusion* can save computation cost through operation sharing and reduce model parameters through weight sharing. From the perspective of metric learning, weight sharing is an essential design to ensure the metric between two branches of data is symmetric. Through *concatenation-based fusion*, we implement this property not only in the feature extraction stage but also in the feature fusion stage. In general, *concatenation-based fusion* improves both efficiency and performance.

Why not window-based self/cross-attention? Since we select stage 3 of the Swin-Transformer as the output, the number of tokens involved is significantly reduced, window-based attention cannot save too many FLOPs. Furthermore, considering the extra latency introduced by the window partition and window reverse operations, window-based attention may even be the slower one.

Why not a query-based decoder? Derivated from vanilla Transformer decoder, many transformer-based models in vision tasks leverage a learnable query to extract the desired objective features from the encoder, like object queries in [3], target query in [46]. However, in our experiment, a query-based decoder suffers from slow convergence and inferior performance. Most Siamese trackers [25, 44, 16] formulate tracking as a foreground-background classification problem, which can better exploit the background information. The vanilla Transformer decoder is a generative model, the generative approaches are considered not suitable for the classification tasks. In another aspect, learning a general target query for any kind of object might cause a bottleneck. In terms of vanilla Transformer encoder-decoder architecture, SwinTrack is an "encoder" only model. Furthermore, quite a little domain knowledge can be easily applied on a classic Siamese tracker to improve the performance, like introducing the smooth movement assumption by using Hanning penalty window on the response map.

Are other forms of motion token feasible? Other forms to construct motion token are possible, such as constructing motion token by summing up the past box coordinate embeddings or representing past object trajectories by one token per box. In our early experiments, we find that the proposed motion token is more effective with the best performance. Summing up the past box coordinate embeddings may result in over-parameterization on the coordinate embeddings. While adding temporal motion representation along with visual features to the single-layer decoder in a multi-token form is ineffective, precise temporal modeling may be required in this form.

3.4 Head and Loss

Head. The head network is split into two branches: classification and bounding box regression. Each of them is a three-layer perceptron. And both of them receives the feature map from the decoder as input to predict the classification response map $r_{cls} \in \mathbb{R}^{(H_x \times W_x) \times 1}$ and bounding box regression map $r_{reg} \in \mathbb{R}^{(H_x \times W_x) \times 4}$, respectively.

Classification loss. In classification branch, we employ the *IoU-aware classification score* as the training target and the *varifocal loss* [50] as the training loss function. IoU-aware design has been very popular recently, but most works consider IoU prediction as an auxiliary branch to assist classification or bounding box regression [52, 2, 44]. To remove the gap between different prediction branches, [50] and [26] replace the hard classification target from the ground-truth value, (i.e., 1 for positive samples, 0 for negative samples), to the IoU between the predicted bounding box and the ground-truth one, which is named the *IoU-aware classification score* (IACS). IACS can help the model select a more accurate bounding box prediction candidate from the pool by trying to predict the quality of the bounding box prediction in another branch at the same position. Along with the IACS, the varifocal loss was proposed in [50] to help the IACS approach outperform other IoU-aware designs.

The classification loss can be formulated as:

$$\mathbb{L}_{cls} = \mathbb{L}_{\mathrm{VFL}}(p, \mathrm{IoU}(b, \hat{b})), \tag{5}$$

where p denotes the predicted IACS, b denotes the predicted bounding box, and $\hat{b}$ denotes the ground-truth bounding box.

Regression loss. For bounding box regression, we employ the generalized IoU loss[35]. The regression loss function can be formulated as:

$$\mathbb{L}_{reg} = \sum_j \mathbb{1}_{\{\mathrm{IoU}(b_j, \hat{b}) > 0\}} [p \mathbb{L}_{\mathrm{GIoU}}(b_j, \hat{b})]. \tag{6}$$

The GIoU loss is weighted by p to emphasize the high classification score samples. The training signals from the negative samples are ignored.

4 Experiments

4.1 Implementation

Model. We design two variants of SwinTrack with different configurations as follows:

- **SwinTrack-T-224**.
  Backbone: Swin Transformer-Tiny [28], pretrained with ImageNet-1k;
  Template size: $[112 \times 112]$; Search region size: $[224 \times 224]$; $C = 384$; $N = 4$;
- **SwinTrack-B-384**.
  Backbone: Swin Transformer-Base [28], pretrained with ImageNet-22k;
  Template size: $[192 \times 192]$; Search region size: $[384 \times 384]$; $C = 512$; $N = 8$;

where C and N are the channel number of the hidden layers in the first stage of Swin Transformer and the number of encoder blocks in feature fusion, respectively. In all variants, we use the output after the third stage of Swin Transformer for feature extraction. Thus, the backbone stride s is 16.

For motion token, the number of sampled object trajectory n is set to 16, the fixed sampling interval Δ is set to 15. If the frame rate of the video sequence is available, the sampling interval is adjusted according to the frame rate. Suppose the frame rate is $\mathbb{f}$, the new sampling interval is getting by $\frac{\Delta}{30}\mathbb{f}$, 30 fps is the standard frame rate we assumed. $\mathbb{g}$, which controls the embedding granularity, is set to the same size as the search region feature map, like 14 for SwinTrack-T-224, and 24 for SwinTrack-B-384. For the model for GOT-10k sequences, n is set to 8, Δ is set to 8, and no frame rate adjustment is applied.

Training. We train SwinTrack using the training splits of LaSOT [11], TrackingNet [33], GOT-10k [19] (1,000 videos are removed following [46] for fair comparison) and COCO 2017 [27]. In addition, we report the results of SwinTrack-T-224 and SwinTrack-B-384 with GOT-10k training split only to follow the protocol described in [19].

The model is optimized with AdamW [29], with a learning rate of 5e-4, and a weight decay of 1e-4. The learning rate of the backbone is set to 5e-5. We train the network on 8 NVIDIA V100 GPUs for 300 epochs with 131,072 samples per epoch. The learning rate is dropped by a factor of 10 after 210 epochs. A 3-epoch linear warmup is applied to stabilize the training process. DropPath [23] is applied on the backbone and the encoder with a rate of 0.1. For the models trained for the GOT-10k evaluation protocol, to prevent over-fitting, the training epoch is set to 150, and the learning rate is dropped after 120 epochs.

For the motion token, the object trajectory for the Siamese training pair is generated with the method described above. The frames that object annotated as absent or out of the video sequence are marked as invalid, the corresponding box coordinates set to $-\infty$. Since the coarse granularity of the coordinate embedding in our setting is already can be seen as an augmentation of historical object trajectory, no additional data augmentation is applied.

Inference. We follow the common procedures for Siamese network-based tracking [1]. The template image is cropped from the first frame of the video sequence. The target object is in the center of the image with a background area factor of 2. The search region is cropped from the current tracking frame, and the image center is the target center position predicted in the previous frame. The background area factor for the search region is 4.

Our SwinTrack takes the template image and search region as inputs and output classification map r_{cls} and regression map r_{reg}. To utilize positional prior in tracking, we apply hanning window penalty on r_{cls}, and the final classification map r'_{cls} is obtained via $r'_{cls} = (1 - \gamma) \times r_{cls} + \gamma \times h$, where γ is the weight parameter and h is the Hanning window with the same size as r_{cls}. The target position is determined by the largest value in r'_{cls} and the scale is estimated based on the corresponding regression results in r_{reg}.

For the motion token, the predicted confidence score and bounding box are collected on the fly. A confidence threshold θ_{conf} is applied, if the confidence score given by the classification branch of the

Table 1: Experiments and comparisons on five benchmarks: LaSOT, LaSOT$_{\text{ext}}$, TrackingNet, GOT-10k and TNL2k.

Tracker	LaSOT [11]		LaSOT$_{\text{ext}}$ [10]		TrackingNet [33]		GOT-10k [19]			TNL2k [42]	
	SUC	P	SUC	P	SUC	P	AO	$SR_{0.5}$	$SR_{0.75}$	SUC	P
C-RPN [12]	45.5	44.3	27.5	32.0	66.9	61.9	-	-	-	-	-
SiamPRN++ [24]	49.6	49.1	34.0	39.6	73.3	69.4	51.7	61.6	32.5	41.3	41.2
Ocean [52]	56.0	56.6	-	-	-	-	61.1	72.1	47.3	38.4	37.7
DiMP [2]	56.9	56.7	39.2	45.1	74.0	68.7	61.1	71.7	49.2	44.7	43.4
LTMU [7]	57.2	57.2	41.4	47.3	-	-	-	-	-	48.5	47.3
SiamR-CNN [39]	64.8	-	-	-	81.2	80.0	64.9	72.8	59.7	52.3	52.8
STMTrack [14]	60.6	63.3	-	-	80.3	76.7	64.2	73.7	57.5	-	-
AutoMatch [51]	58.3	59.9	37.6	43.0	76.0	72.6	65.2	76.6	54.3	-	-
TrDiMP [40]	63.9	61.4	-	-	78.4	73.1	67.1	77.7	58.3	-	-
TransT [5]	64.9	69.0	-	-	81.4	80.3	67.1	76.8	60.9	51.0	-
STARK [46]	67.1	-	-	-	82.0	-	68.8	78.1	64.1	-	-
KeepTrack [31]	67.1	70.2	48.2	-	-	-	-	-	-	-	-
SwinTrack-T-224	67.2	70.8	47.6	53.9	81.1	78.4	71.3	81.9	64.5	53.0	53.2
SwinTrack-B-384	71.3	76.5	49.1	55.6	84.0	82.8	72.4	80.5	67.8	55.9	57.1

head is lower than the threshold, the target object in the current frame is marked as lost by setting the collected bounding box to $-\infty$. θ_{conf} is set to 0.4 for LaSOT, the rests are set to 0.3.

4.2 Comparisons to State-of-the-arts

We conduct experiments and compare SwinTrack with SoTA trackers on five benchmarks.

LaSOT. LaSOT [11] consists of 280 videos for test. Tab. 1 shows the results and comparisons with SoTAs. From Tab. 1, we can observe that SwinTrack-T-224 with light architecture reaches SoTA performance with 0.672 SUC and 0.708 PRE scores, which is competitive compared with other Transformer-based trackers, including STARK-ST101 (0.671 SUC score) and TransT (0.649 SUC), and other trackers using complicated designs such as KeepTrack (0.671 SUC) and SiamR-CNN (0.648 SUC score). With a larger backbone and input size, our strongest variant SwinTrack-B-384 sets a new record with 0.713 SUC score, surpassing START-ST101 and KeepTrack by 4.2 absolute percentage points.

LaSOT$_{\text{ext}}$. The recent LaSOT$_{\text{ext}}$ [10] is an extension of LaSOT by adding 150 extra videos. These new sequences are challenging as many similar distractors cause difficulties for tracking. The results of our tracker related to this dataset are an average of three times. KeepTrack uses a complex association technique to handle distractors and achieves a promising 0.482 SUC score as in Tab. 1. Compared with complicated KeepTrack, SwinTrack-T-224 is simple and neat, yet shows comparable performance with 0.476 SUC score. In addition, due to complicated design, KeepTrack runs at less than 20 *fps*, while SwinTrack-T-224 runs in 98 *fps*, 5$\times$ faster than KeepTrack. When using a larger model, SwinTrack-B-384 shows the best performance with 0.491 SUC score.

TrackingNet. We evaluate different trackers on the test set of TrackingNet [33]. From Tab. 1, we observe that our SwinTrack-T-224 achieves a comparable result with 0.811 SUC score. Using a larger model and input size, SwinTrack-B-384 obtains the best performance with 0.840 SUC score, better than STARK-ST101 with 0.820 SUC score and TransT with 0.814 SUC score.

GOT-10k. GOT-10k [19] offers 180 videos for test and it requires trackers to be trained using GOT-10k train split only. From Tab. 1, we see that SwinTrack-B-384 achieves the best mAO of 0.724, and SwinTrack-T-224 obtains a mAO of 0.713. Both models outperform other Transformer-based counterparts significantly, including START-ST101 (0.688 mAO), TransT (0.671 mAO) and TrDiMP (0.671 mAO).

TNL2k. TNL2k [42] is a newly released tracking dataset with 700 videos for test. As reported in Tab. 1, both models surpass the others. SwinTrack-B-384 set a new state-of-the-art with 0.559 SUC score.

Efficiency comparison. We report the comparisons of SwinTrack with other Transformer-based trackers in terms of efficiency and complexity. As displayed in Tab. 2, SwinTrack-T-224 with a small model runs the fastest with a speed of *98 fps*. Especially, compared with STARK-ST101 and

Table 2: Comparison on running speed and # parameters with other Transformer-based trackers.

Tracker	Speed (*fps*)	MACs[2] (G)	Params (M)
TrDiMP [40]	26	-	-
TransT [5]	50	-	23
STARK-ST50 [46]	42	10.9	24
STARK-ST101 [46]	32	18.5	42
SwinTrack-T-224	98	6.4	23
SwinTrack-B-384	45	69.7	91

Table 3: Ablation experiments of SwinTrack on four benchmarks. The experiments are conducted on SwinTrack-T-224 without the motion token. ❶: baseline method, *i.e.*, SwinTrack-T-224 without motion token; ❷: replacing Transformer backbone in the baseline method with ResNet-50; ❸: replacing our feature fusion with cross attention-based fusion in the baseline method; ❹: replacing the decoder in baseline with a target query-based; ❺: replacing united positional encoding with absolute sine position encoding in the baseline method; ❻: replacing the IoU-aware classification loss with the plain binary cross entropy loss; ❼: removing the Hanning penalty window in the baseline method inference.

	LaSOT SUC (%)	$\text{LaSOT}_{\text{ext}}$ SUC (%)	TrackingNet SUC (%)	GOT-10k[3] mAO (%)	Speed *fps*	Params M
❶	66.7	46.9	80.8	70.9	98	22.7
❷	64.2	41.8	79.5	68.2	121	20.0
❸	66.6	45.4	80.2	69.3	72	34.6
❹	66.6	43.2	79.6	69.0	91	25.3
❺	65.7	45.0	80.0	70.0	103	21.6
❻	66.2	46.7	79.4	68.2	98	22.7
❼	65.7	46.0	80.0	69.6	98	22.7

STARK-ST50 with 32 *fps* and 42 *fps*, SwinTrack-T-224 is 3× and 2× faster. Despite using a larger model, our SwinTrack-B-384 is still faster than STARK-ST101 and STARK-ST50.

4.3 Ablation Experiment

Comparison with ResNet backbone. We compare the Swin-Transformer backbone with popular ResNet-50 [18]. As shown in Tab. 3 (❶ *vs.* ❷). The Swin Transformer backbone significantly boosts the performance by 2.5% SUC score in LaSOT, 5.1% SUC score in $\text{LaSOT}_{\text{ext}}$. The result shows that the strong appearance modeling capability provided by the Swin Transformer plays a crucial role.

Feature fusion. As displayed in Tab. 3 (❶ *vs.* ❸), compared with the *concatenation-based fusion*, the *cross attention-based fusion* runs at a slower speed, occupies much more memory, and also has an inferior performance on all datasets. Slower speed can be due to the latency brought by the extra operations. The parameter-sharing strategy not only just reduces the number of parameters but also benefits metric learning.

Comparison with the query-based decoder. Queries is commonly adopted in the decoder of Transformer network in vision tasks, e.g. object query [3] and target query [46]. Nevertheless, our empirical results in Tab. 3 (❶ *vs.* ❹) show that a target query-based decoder degrades the tracking performance on all benchmarks, even with 2× training pairs. As discussed, one possible reason is the generative model is not suitable for classification. Besides, learning a general target query for any kind of object may also be difficult.

Position encoding. We compare the united positional encoding used in SwinTrack and the original absolute position encoding in Transformer [38]. Notice, We make a little modification to the original absolute position encoding: Except for the 2D embedding, the index of token source (e.g. 1 for the tokens from the template patch, 2 for the tokens from the search region patch) is also embedded. As

[2]Multiply–accumulate operation

[3]The GOT-10k results in this column are trained with full training datasets.

Table 4: Ablation experiments on our proposed motion token on the tracking performance on four benchmarks. The experiments are conducted on SwinTrack-T-224. ❶: SwinTrack-T-224; ❷: SwinTrack-B-384; ❸: SwinTrack-T-224 without motion token; ❹: SwinTrack-B-384 without motion token; ❺: replacing the motion token in SwinTrack-T-224 with a learnable embedding token.

	LaSOT	LaSOT$_{\text{ext}}$	TrackingNet	GOT-10k	Speed
	SUC (%)	SUC (%)	SUC (%)	mAO (%)	*fps*
❶	67.2	47.6	81.1	71.3	96
❷	71.3	49.1	84.0	72.4	45
❸	66.7	47.0	80.8	70.0	98
❹	70.2	48.5	84.0	70.7	45
❺	66.3	45.2	81.2	70.0	96

shown in Tab. 3 (❶ *vs.* ❺), our method with united positional encoding obtains improvements with 0.8-1.9 absolute percentage points on the benchmarks with negligible loss in speed (98 *vs.* 103).

Loss function. From Tab. 3 (❶ *vs.* ❻), we observe that the model trained with varifocal loss significantly outperforms the one with binary cross entropy (BCE) loss without loss of efficiency. This result indicates that the varifocal loss can assist the classification branch of the head to generate an IoU-aware response map, and thus help the tracker to improve the tracking performance.

Post processing. One may wonder with highly discriminative Transformer architecture and IoU-aware classification loss does the hanning penalty window is still functional, which introduces a strong smooth movement assumption. In the experiments, we remove the hanning penalty window in post-processing, as shown in Tab. 3 (❶ *vs.* ❼), the performance is dropped by 1.0 SUC for LaSOT, 1.3 AO for GOT-10k in absolute percentage, and less than 1% in the SUC metric of other datasets. This suggests that the strong smooth movement assumption is still applicable for our tracker. But compared with the former Transformer-based tracker [5], the performance gap between with and without penalty window post-processing is narrowing.

Effectiveness of motion token. We study the effectiveness of the motion token by conducting comparison experiments. As shown in Tab. 4 (❶ vs. ❸ and ❷ vs. ❹), the models with motion token outperforms the models without motion token on all datasets, especially on LaSOT$_{\text{ext}}$ and GOT-10k. The results indicate that the motion token can assist the tracker to handle hard similar distractors in LaSOT$_{\text{ext}}$ and stabilize the short-term tracking like the sequences in GOT-10k test set. We also study whether the effectiveness of the motion token is simply from the extra embedding vector. We set up an experiment as in Tab. 4 (❺), which replaces the motion token with a learnable embedding token. The result shows that the extra embedding vector has negative impacts indicating the effectiveness of the embedding of object trajectory.

5 Conclusion

In this work, we present SwinTrack, a simple and strong baseline for Transformer tracking. In SwinTrack, both representation learning and feature fusion are implemented with the attention mechanism. Extensive experiments demonstrate the effectiveness of such architecture. Besides, we propose the motion token to enhance the robustness of the tracker by providing the historical object trajectory, showing the flexibility of the Transformer model in architectural design. With the power of sequence-to-sequence model architecture, a context-rich tracker is possible, and more contextual cues can be incorporated. Finally, We hope this work can inspire and facilitate future research.

Acknowledgments and Disclosure of Funding

This work is supported by Peng Cheng Laboratory Research Project No. PCL2021A07. Heng Fan and his employer received no financial support for the research, authorship, and/or publication of this article.

References

[1] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H., 2016. Fully-convolutional siamese networks for object tracking, in: ECCVW.

[2] Bhat, G., Danelljan, M., Gool, L.V., Timofte, R., 2019. Learning discriminative model prediction for tracking, in: ICCV.

[3] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers, in: ECCV.

[4] Chen, C.F.R., Fan, Q., Panda, R., 2021a. Crossvit: Cross-attention multi-scale vision transformer for image classification, in: ICCV.

[5] Chen, X., Yan, B., Zhu, J., Wang, D., Yang, X., Lu, H., 2021b. Transformer tracking, in: CVPR.

[6] Cui, Y., Jiang, C., Wang, L., Wu, G., 2022. Mixformer: End-to-end tracking with iterative mixed attention, in: CVPR.

[7] Dai, K., Zhang, Y., Wang, D., Li, J., Lu, H., Yang, X., 2020. High-performance long-term tracking with meta-updater, in: CVPR.

[8] Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M., 2019. Atom: Accurate tracking by overlap maximization, in: CVPR.

[9] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2021. An image is worth 16x16 words: Transformers for image recognition at scale, in: ICLR.

[10] Fan, H., Bai, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Huang, M., Liu, J., Xu, Y., et al., 2021. Lasot: A high-quality large-scale single object tracking benchmark. International Journal of Computer Vision 129, 439–461.

[11] Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H., 2019. Lasot: A high-quality benchmark for large-scale single object tracking, in: CVPR.

[12] Fan, H., Ling, H., 2019. Siamese cascaded region proposal networks for real-time visual tracking, in: CVPR.

[13] Fan, H., Ling, H., 2021. Cract: Cascaded regression-align-classification for robust visual tracking, in: IROS.

[14] Fu, Z., Liu, Q., Fu, Z., Wang, Y., 2021. Stmtrack: Template-free visual tracking with space-time memory networks, in: CVPR.

[15] Gao, S., Zhou, C., Ma, C., Wang, X., Yuan, J., 2022. Aiatrack: Attention in attention for transformer visual tracking .

[16] Han, W., Dong, X., Khan, F.S., Shao, L., Shen, J., 2021. Learning to fuse asymmetric feature maps in siamese trackers, in: CVPR.

[17] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R., 2022. Masked autoencoders are scalable vision learners, in: CVPR.

[18] He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: CVPR.

[19] Huang, L., Zhao, X., Huang, K., 2019. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence 43, 1562–1577.

[20] Ke, G., He, D., Liu, T.Y., 2021. Rethinking positional encoding in language pre-training, in: ICLR.

[21] Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L., 2016. A novel performance evaluation methodology for single-target trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence 38, 2137–2155.

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. NIPS .

[23] Larsson, G., Maire, M., Shakhnarovich, G., 2016. Fractalnet: Ultra-deep neural networks without residuals, in: ICLR.

[24] Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.S., 2019. Evolution of siamese visual tracking with very deep networks, in: CVPR.

[25] Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X., 2018. High performance visual tracking with siamese region proposal network, in: CVPR.

[26] Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J., 2020. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, in: NeurIPS.

[27] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: ECCV.

[28] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. ICCV .

[29] Loshchilov, I., Hutter, F., 2019. Decoupled weight decay regularization, in: ICLR.

[30] Mayer, C., Danelljan, M., Bhat, G., Paul, M., Paudel, D.P., Yu, F., Van Gool, L., 2022. Transforming model prediction for tracking, in: CVPR.

[31] Mayer, C., Danelljan, M., Paudel, D.P., Van Gool, L., 2021. Learning target candidate association to keep track of what not to track, in: ICCV.

[32] Mueller, M., Smith, N., Ghanem, B., 2016. A benchmark and simulator for uav tracking, in: ECCV.

[33] Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B., 2018. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild, in: ECCV.

[34] Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks, in: NIPS.

[35] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S., 2019. Generalized intersection over union .

[36] Shaw, P., Uszkoreit, J., Vaswani, A., 2018. Self-attention with relative position representations. arXiv .

[37] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H., 2021. Training data-efficient image transformers & distillation through attention, in: ICML.

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: NeurIPS.

[39] Voigtlaender, P., Luiten, J., Torr, P.H., Leibe, B., 2020. Siam r-cnn: Visual tracking by re-detection, in: CVPR.

[40] Wang, N., Zhou, W., Wang, J., Li, H., 2021a. Transformer meets tracker: Exploiting temporal context for robust visual tracking, in: CVPR.

[41] Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L., 2021b. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions, in: ICCV.

[42] Wang, X., Shu, X., Zhang, Z., Jiang, B., Wang, Y., Tian, Y., Wu, F., 2021c. Towards more flexible and accurate object tracking with natural language: Algorithms and benchmark, in: CVPR.

[43] Xie, F., Wang, C., Wang, G., Cao, Y., Yang, W., Zeng, W., 2022. Correlation-aware deep tracking, in: CVPR.

[44] Xu, Y., Wang, Z., Li, Z., Yuan, Y., Yu, G., 2020. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines, in: AAAI.

[45] Yan, B., Jiang, Y., Sun, P., Wang, D., Yuan, Z., Luo, P., Lu, H., 2022. Towards grand unification of object tracking, in: ECCV.

[46] Yan, B., Peng, H., Fu, J., Wang, D., Lu, H., 2021. Learning spatio-temporal transformer for visual tracking, in: ICCV.

[47] Ye, B., Chang, H., Ma, B., Shan, S., Chen, X., 2022. Joint feature learning and relation modeling for tracking: A one-stream framework .

[48] Yu, Y., Xiong, Y., Huang, W., Scott, M.R., 2020. Deformable siamese attention networks for visual object tracking, in: CVPR.

[49] Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S., 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet, in: ICCV.

[50] Zhang, H., Wang, Y., Dayoub, F., Sünderhauf, N., 2021a. Varifocalnet: An iou-aware dense object detector, in: CVPR.

[51] Zhang, Z., Liu, Y., Wang, X., Li, B., Hu, W., 2021b. Learn to match: Automatic matching network design for visual tracking, in: ICCV.

[52] Zhang, Z., Peng, H., Fu, J., Li, B., Hu, W., 2020. Ocean: Object-aware anchor-free tracking, in: ECCV.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We observed unstable evaluation results on LaSOT$_{\text{ext}}$ dataset, so the results from this dataset are averaged from three times run. We didn't observe such phenomenon on other datasets.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] All are the commonly used public datasets.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Positional Encoding

Transformer requires a positional encoding to identify the position of the current processing token [38]. Through a series of comparison experiments, we choose *untied positional encoding*, which is proposed in TUPE [20], as the positional encoding solution of our tracker. In addition, we generalize the *untied positional encoding* to arbitrary dimensions to fit with other components in our tracker.

The original transformer [38] proposes a absolute positional encoding method to represent the position: a fixed or learnable vector p_i is assigned to each position i. Starting from the basic attention module, we have:

$$\mathrm{Atten}(Q,K,V) = \mathrm{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}V\Big), \tag{7}$$

where Q,K,V are the $query$ vector, key vector and $value$ vector, which are the parameters of the attention function, d_k is the dimension of key. Introducing the linear projection matrix and multi-head attention to the attention module (7), we get the multi-head variant defined in [38]:

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W_O, \tag{8}$$

where $\mathrm{head_i} = \mathrm{Atten}(QW_i^Q, KW_i^K, VW_i^V)$, $W_i^Q \in \mathbb{R}^{d_\mathrm{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_\mathrm{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_\mathrm{model}\times d_v}$, $W_i^O \in \mathbb{R}^{hd_v \times d_\mathrm{model}}$ and h is the number of heads. For simplicity, as in [20], we assume that $d_k = d_v = d_\mathrm{model}$, and use the single-head version of self-attention module. Denoting the input sequence as $x = x_1, x_2, \ldots, x_n$, where n is the length of sequence, x_i is the i-th token in the input data. Denoting the output sequence as $z = (z_1, z_2, \ldots, z_n)$. Self-attention module can be rewritten as

$$z_i = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n}\exp(\alpha_{ij'})}(x_j W^V), \tag{9}$$

$$\text{where } \alpha_{ij} = \frac{1}{\sqrt{d}}(x_i W^Q)(x_j W^K)^T. \tag{10}$$

Obviously, the self-attention module is permutation-invariance. Thus it can not "understand" the order of input tokens.

Untied absolute positional encoding. By adding a learnable positional encoding [38] to the single-head self-attention module, we can obtain the following equation:

$$\begin{aligned}\alpha_{ij}^{Abs} &= \frac{((w_i+p_i)W^Q)((w_j+p_j)W^K)^T}{\sqrt{d}} \\ &= \frac{(w_iW^Q)(w_jW^K)^T}{\sqrt{d}} + \frac{(w_iW^Q)(p_jW^K)^T}{\sqrt{d}} \\ &+ \frac{(p_iW^Q)(w_jW^K)^T}{\sqrt{d}} + \frac{(p_iW^Q)(p_jW^K)^T}{\sqrt{d}}.\end{aligned} \tag{11}$$

The equation (11) is expanded into four terms: token-to-token, token-to-position, position-to-token, position-to-position. [20] discuss the problems that exist in the equation and proposes the *untied absolute positional encoding*, which unties the correlation between tokens and positions by removing the token-position correlation terms in equation (11), and using an isolated pair of projection matrices U^Q and U^K to perform linear transformation upon positional embedding vector. The following is the new formula for obtaining α_{ij} using the *untied absolute positional encoding* in the l-th layer:

$$\begin{aligned}\alpha_{ij} &= \frac{1}{\sqrt{2d}}(x_i^l W^{Q,l})(x_j^l W^{K,l})^T \\ &+ \frac{1}{\sqrt{2d}}(p_i U^Q)(p_j U^K)^T.\end{aligned} \tag{12}$$

where p_i and p_j is the positional embedding at position i and j respectively, $U^Q \in \mathbb{R}^{d\times d}$ and $U^K \in \mathbb{R}^{d\times d}$ are learnable projection matrices for the positional embedding vector. When extending to the multi-head version, the positional embedding p_i is shared across different heads, while U^Q and U^K are different for each head.

Relative positional bias. According to [36], relative positional encoding is a necessary supplement to absolute positional encoding. In [20], a relative positional encoding is applied by adding a relative positional bias to equation (12):

$$\begin{aligned}\alpha_{ij} &= \frac{1}{\sqrt{2d}}(x_i^l W^{Q,l})(x_j^l W^{K,l})^T \\ &+ \frac{1}{\sqrt{2d}}(p_i U^Q)(p_j U^K)^T + b_{j-i},\end{aligned} \tag{13}$$

where for each $j-i$, b_{j-i} is a learnable scalar. The *relative positional bias* is also shared across layers. When extending to the multi-head version, b_{j-i} is different for each head.

Generalize to multiple dimensions. Before working with our tracker's encoder and decoder network, we need to extend the *untied positional encoding* to a multi-dimensional version. One straightforward method is allocating a positional embedding matrix for every dimension and summing up all embedding vectors from different dimensions at the corresponding index to represent the final embedding vector. Together with *relative positional bias*, for an n-dimensional case, we have:

$$\begin{aligned}\alpha_{\underbrace{ij\ldots}_{\mathtt{n}},\underbrace{mn\ldots}_{\mathtt{n}}} &= \frac{1}{\sqrt{2d}}(x_{\underbrace{ij\ldots}_{\mathtt{n}}} W^Q)(x_{\underbrace{mn\ldots}_{\mathtt{n}}} W^K)^T \\ &+ \frac{1}{\sqrt{2d}}[\underbrace{(p_i^1 + p_j^2 + \ldots)}_{\mathtt{n}} U^Q][\underbrace{(p_m^1 + p_n^2 + \ldots)}_{\mathtt{n}} U^K]^T \\ &+ b_{\underbrace{m-i, n-j, \ldots}_{\mathtt{n}}}.\end{aligned} \tag{14}$$

Generalize to concatenation-based fusion. In order to work with *concatenation-based fusion*, the *untied absolute positional encoding* is also concatenated to match the real position, the indexing tuple of *relative positional bias* now appends with a pair of indices to reflect the origination of $query$ and key involved currently.

Take l-th layer in the encoder as the example:

$$\begin{aligned}\alpha_{ij,mn,g,h} &= \frac{1}{\sqrt{2d}}(x_{ij,g}^l W^{Q,l})(x_{mn,h}^l W^{K,l})^T \\ &+ \frac{1}{\sqrt{2d}}[(p_{i,g}^1 + p_{j,g}^2)U_g^Q][(p_{m,h}^1 + p_{n,h}^2)U_h^K]^T \\ &+ b_{m-i,n-j,g,h}\ ,\end{aligned} \tag{15}$$

where g and h are the index of the origination of $query$ and key respectively, for instance, 1 for the tokens from the template image, 2 for the tokens from the search image. The form in the decoder is similar, except that g is fixed. In our implementation, the parameters of *untied positional encoding* are shared inside the encoder and the decoder, respectively.

B The Effect of Pre-training Datasets

The two variants of our tracker, SwinTrack-T-224 and SwinTrack-B-384 are using different pre-training datasets, which are derived from the settings from Swin Transformer [28]. Specifically, SwinTrack-T-224 adopts ImageNet-1k and SwinTrack-B-384 adopts ImageNet-22k.

To analyze the effect of different pre-training datasets, we conduct an experiment on the performance of our tracker with different pre-training datasets. Other than the pre-training datasets, The experiment follows the same settings in the ablation study in the paper, the motion token is not used and the results on GOT-10k are trained on the full datasets as described in the paper. From Tab. 5, we can observe that, for smaller model SwinTrack-T-224 (23M # parameters), pre-training on ImageNet-22k brings small improvements on LaSOT (+0.6%) and TrackingNet (+0.4%) but degrades the performance on

Table 5: The effect of Imagenet-22k pre-training. The results are following the settings in the ablation study in the paper (motion token is not used and the result on GOT-10k is trained on the full dataset).

Trackers	Pre-training	LaSOT [11]		LaSOT$_{ext}$ [10]		TrackingNet [33]		GOT-10k [19]		
		SUC	P	SUC	P	SUC	P	AO	SR$_{0.5}$	SR$_{0.75}$
SwinTrack-T-224	ImageNet-1k	66.7	70.6	46.9	52.9	86.7	80.1	69.7	79.0	65.6
SwinTrack-T-224	ImageNet-22k	67.3	71.7	46.0	51.7	81.2	78.9	69.5	78.9	65.5
SwinTrack-B-384	ImageNet-1k	68.0	72.5	47.3	53.2	83.8	82.9	71.8	80.2	67.1
SwinTrack-B-384	ImageNet-22k	70.2	75.3	47.5	53.3	86.9	80.1	70.2	80.7	65.4

Table 6: Performance comparisons with newly released Transformer-based Trackers on four benchmarks: LaSOT, LaSOT$_{ext}$, TrackingNet and GOT-10k.

Tracker	Pre-training	LaSOT [11]		LaSOT$_{ext}$ [10]		TrackingNet [33]		GOT-10k [19]		
		SUC	P	SUC	P	SUC	P	AO	SR$_{0.5}$	SR$_{0.75}$
STARK [46]	ImageNet-1k	67.1	-	-	-	82.0	-	68.8	78.1	64.1
SBT [43]	ImageNet-1k	66.7	71.1	-	-	-	-	70.4	80.8	64.7
ToMP [30]	ImageNet-1k	68.5	73.5	45.9	-	81.5	78.9	-	-	-
MixFormer [6]	ImageNet-22k	70.1	76.3	-	-	83.9	83.1	-	-	-
AiATrack [15]	ImageNet-1k	69.0	73.8	47.7	55.4	82.7	80.4	69.6	80.0	63.2
Unicorn [45]	ImageNet-1k	68.5	74.1	-	-	83.0	82.2	-	-	-
OSTrack [47]	MAE [17]	71.1	77.6	50.5	57.6	83.9	83.2	73.7	83.2	70.8
SwinTrack-T-224	ImageNet-1k	67.2	70.8	47.6	53.9	81.1	78.4	71.3	81.9	64.5
SwinTrack-B-384	ImageNet-22k	71.3	76.5	49.1	55.6	84.0	82.8	72.4	80.5	67.8

GOT-10k (-1.4%). For larger model SwinTrack-B-384 (91M # parameters), pre-training on ImageNet-22k shows significant performance gains on LaSOT (+2.2%) and GOT-10k (+3.0%) but slightly degrades the result on TrackingNet (-0.6%). On LaSOT$_{ext}$, ImageNet-22k shows a performance degradation on smaller model SwinTrack-T-224 (-0.9%) and brings small improvements on larger model SwinTrack-B-384 (+0.2%).

C Comparison with Newly Released Transformer-based Trackers

We compare our tracker with some newly released Transformer-based trackers, including STARK [46], SBT [43], ToMP [30], MixFormer [6], AiATrack [15], Unicorn [45], OSTrack [47] in Tab. 6 in four challenging benchmarks. The result shows our tracker is still competitive.

Fig. 3 and Fig. 4 show the success plot and the precision plot respectively. The comparison includes our SwinTrack-T-224, our SwinTrack-B-384, TransT[5], STARK[46], MixFormer[6], AiATrack[15] and ToMP[30]. Our tracker obtained the best performance on this benchmark. By looking into the curves of the figures, there is a significant advantage in the bounding box accuracy compared with other trackers due to our fully attentional architecture.

The success AUC score under different attributes of LaSOT [11] Test set in shown in Fig. 5. Fig. 5 indicates that our tracker has no obvious shortcomings except the viewpoint change.

D Results on UAV123 and VOT Benchmark

In this section, we report the performance of the tracker on three additional benchmarks, including UAV123 [32], VOT2020 and VOT-STB2022 [21].

UAV123[32] is an aerial video dataset and benchmark for low-altitude UAV target tracking, containing 123 video sequences. Our tracker is on par with the state-of-the-art, AiATrack [15], on this benchmark. The results are shown in Tab. 7.

Finally, we evaluate our tracker on the two versions of the VOT Challenge: VOT2020 and VOT-STB2022. The VOT2020 dataset contains 60 videos with segmentation masks annotated. Since our tracker is a bounding box only method, we compare the results with the trackers that produce the bounding boxes as well. The result in Tab.8 shows that SwinTrack-T-224 has a better performance than the larger SwinTrack-B-384 on this benchmark.

Table 7: Comparison to the state-of-the-arts on UAV123 [32] benchmark.

	Ocean [52]	DiMP [2]	TransT [5]	ToMP 50[30]	MixFormer 22k[6]	AiATrack [15]	SwinTrack T-224	SwinTrack B-384
AUC (%)	62.1	65.3	69.1	69.0	70.4	70.6	68.8	70.5

Table 8: Comparison to the state-of-the-art bounding box only methods on VOT2020ST [21].

	ATOM [8]	DiMP [2]	STARK 50[46]	STARK 101[46]	ToMP 50[30]	ToMP 101[30]	SwinTrack T-224	SwinTrack B-384
EAO	0.271	0.274	0.308	0.303	0.297	0.309	0.302	0.283
Accuracy	0.462	0.457	0.478	0.481	0.453	0.453	0.471	0.472
Robustness	0.734	0.734	0.799	0.775	0.789	0.814	0.775	0.741

In addition, We report the results on VOT-STB2022 in Tab.9. SwinTrack-T-224 has a better performance on VOT-STB2022 as well. No comparison is made since VOT-STB2022 is a newly released benchmark.

E Quantitative Analysis of the Effectiveness of Motion Token

To give a further analysis of the effectiveness of motion token, we provide the success plot (Fig. 6) and precision plot (Fig. 7) on LaSOT test set, and the success AUC score under different attributes of LaSOT test set in Fig. 8. The success plot and the precision plot show that the motion token improves the performance of the trackers by boosting robustness. While the Fig. 8 further points out that the motion token can assist the tracker to recover from a failure state when the vision features are not reliable like an object is getting out of view or fully occluded by other objects.

F Response Visualization for Qualitative Analysis

We provide the heatmap visualization of the response map generated by the IoU-aware classification branch of the head in our SwinTrack-B-384 in Fig. 9. The visualized sequences are from LaSOT$_{\text{ext}}$ [10], with challenges include fast motion, full occlusion, hard distractor, *etc*. The results demonstrate the great discriminative power of our tracker. Many trackers will show a multi-peak on the response map when the target object is occluded or multiple similar objects exist. With the vision-motion integrated Transformer architecture, our tracker eases such phenomenon.

G Failure Case

We show some typical failure cases of our tracker (SwinTrack-B-384 on LaSOT$_{\text{ext}}$ [10] and VOT-STB2022 [21]) in Fig. 10. The first case suffers from a mixture of low resolution, fast motion, and background clutter. The second case suffers from a fast occlusion by a distractor. The third case suffers from the non-semantic target.

Table 9: Results on VOT-STB2022 [21].

	SwinTrack T-224	SwinTrack B-384
EAO	0.505	0.477
Accuracy	0.777	0.790
Robustness	0.790	0.759

Figure 3: Comparison to the state-of-the-art trackers on LaSOT [11] Test set using success (SUC) AUC score.

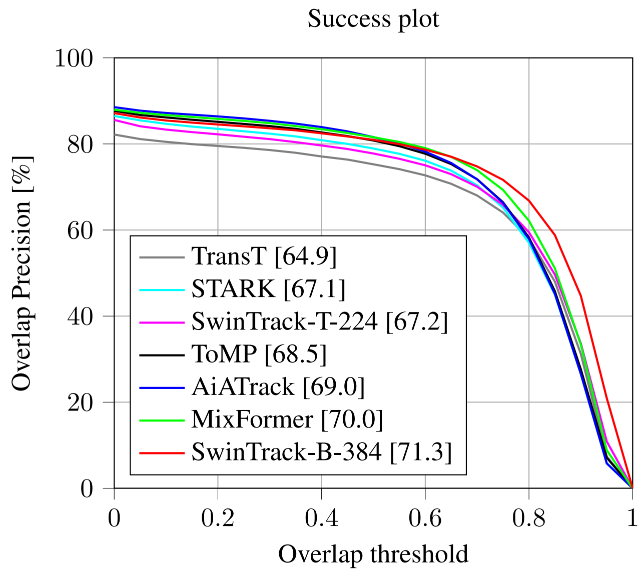


Figure 4: Comparison to the state-of-the-art trackers on LaSOT [11] Test set using precision (PRE) AUC score.

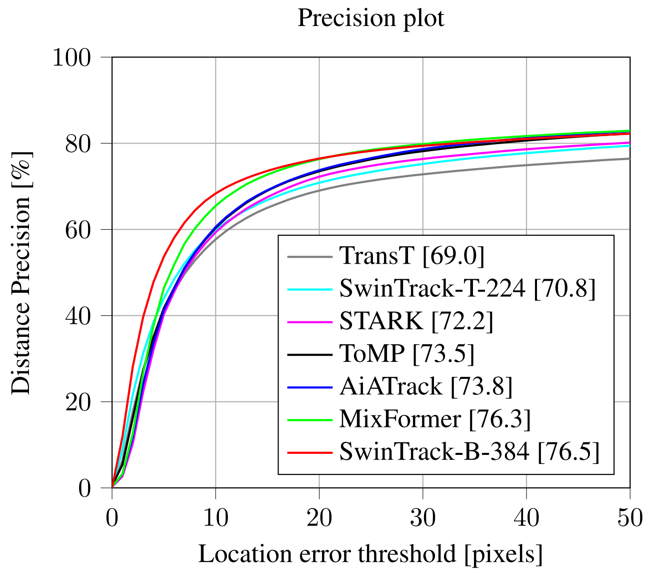


Figure 5: Comparison to the state-of-the-art trackers using success (SUC) AUC score under different attributes of LaSOT [11] Test set.

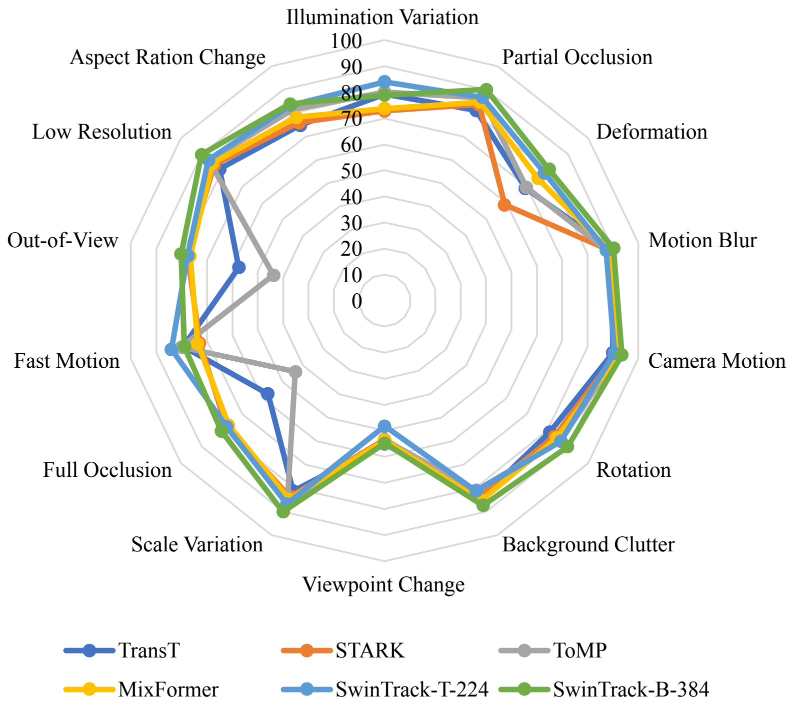


Figure 6: Success (SUC) AUC score on LaSOT [11] Test set assessing the effectiveness of the motion token.

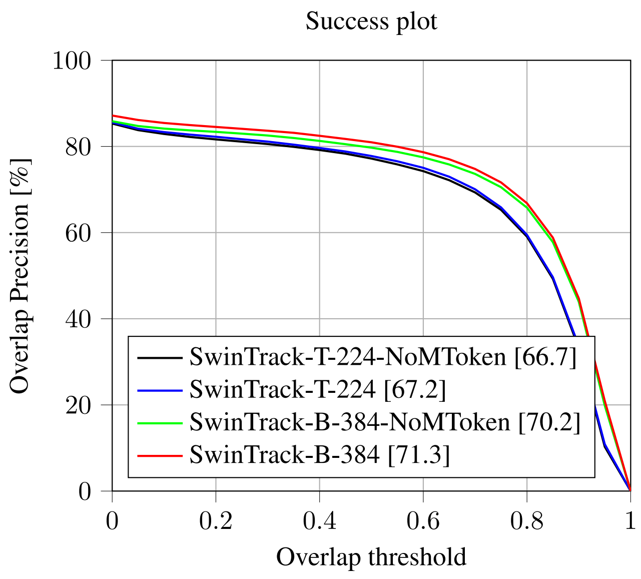


Figure 7: Precision (PRE) AUC score on LaSOT [11] Test set assessing the effectiveness of the motion token.

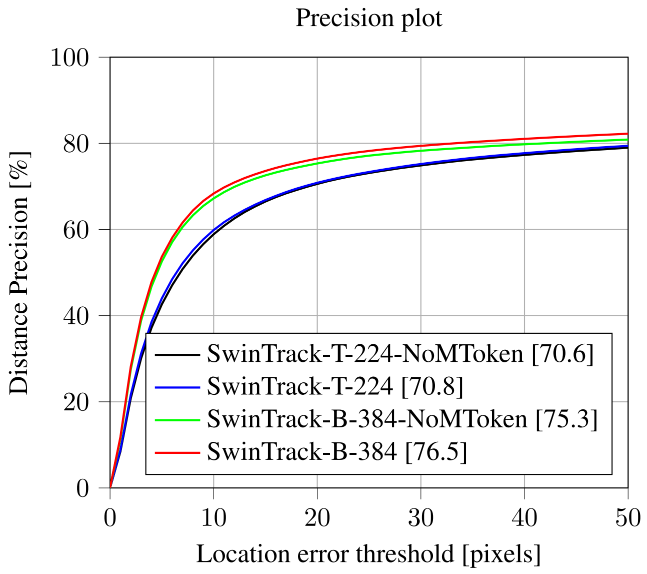


Figure 8: Success (SUC) AUC score under different attributes of LaSOT [11] Test set assessing the effectiveness of the motion token.

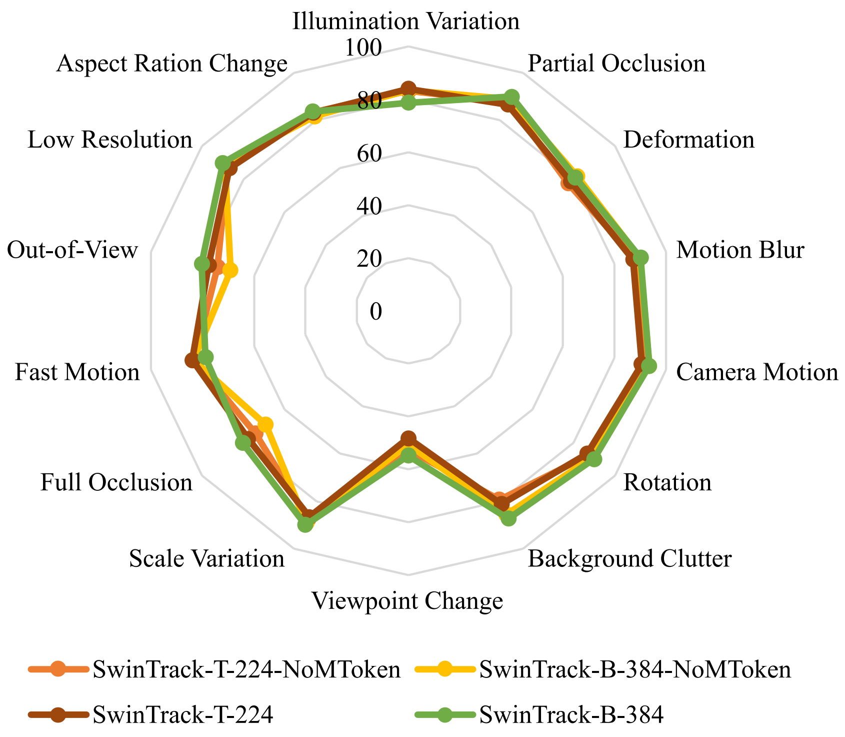


Figure 9: Heatmap visualization of the tracking response map of our SwinTrack-B-384 on LaSOT$_{\text{ext}}$ [10]. The odd rows visualize the search region patches with ground-truth bounding box (in red rectangles). The even rows visualize the search region patches blended with the heatmap visualization of the response map. The sequences and challenges involved: atv-10 (POC, ROT, VC, SV, LR, ARC), wingsuit-10 (CM, BC, VC, SV, FOC, LR, ARC), rhino-9 (DEF, SV, ARC) and misc-3 (POC, MB, ROT, BC, SV, FOC, FM, LR).

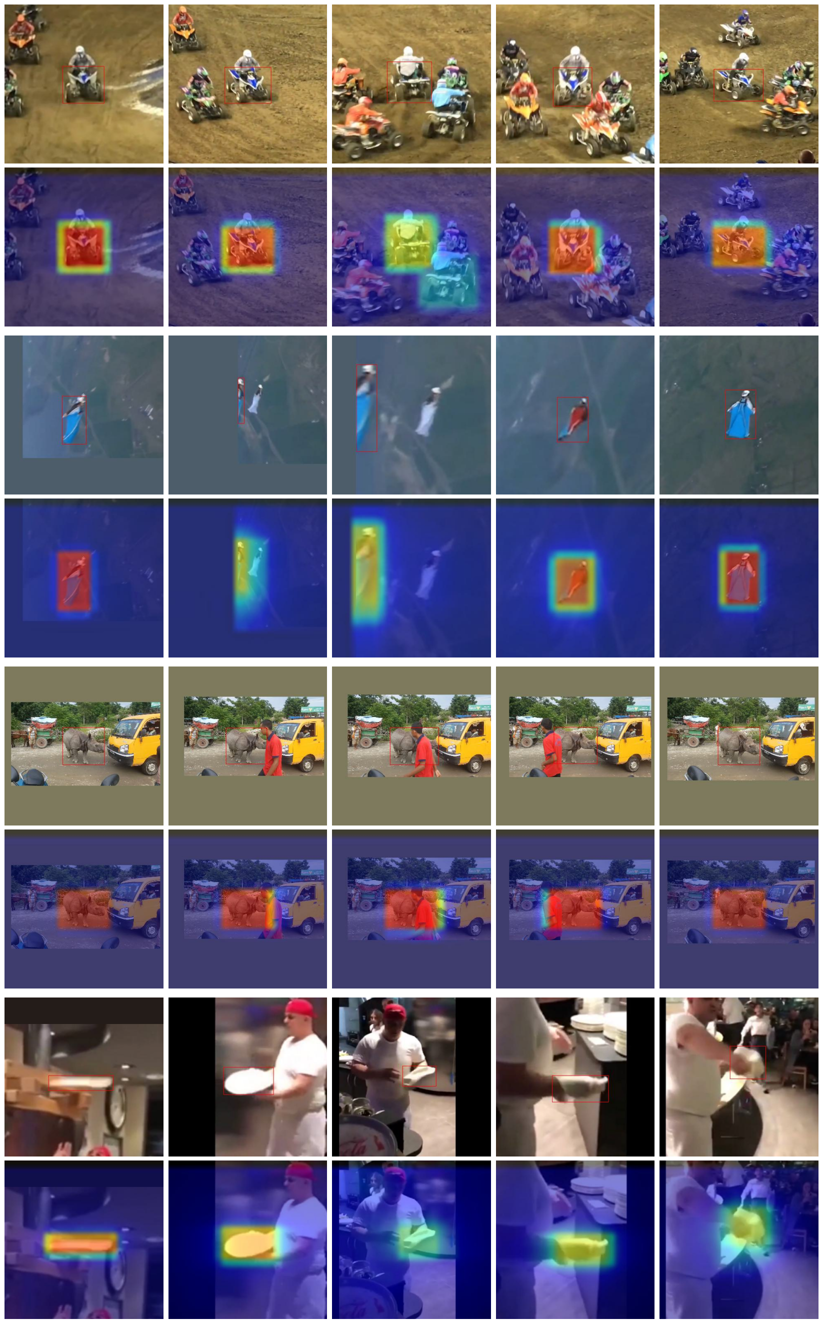
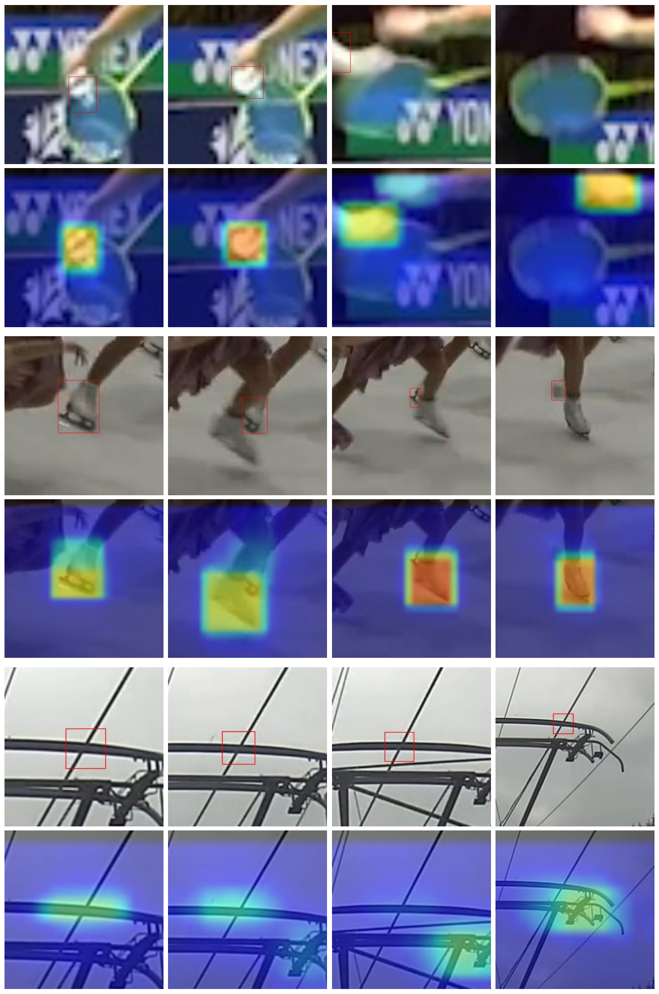


Figure 10: Heatmap visualization of the failure cases. The organizational form is the same as Fig. 9. The sequences and challenges involved: badminton-3 in LaSOT$_{ext}$ (MB, SV, FOC, FM, OV, LR, ARC), skatingshoe-2 in LaSOT$_{ext}$ (POC, MB, ROT, BC, SV, FOC, FM, LR, ARC) and conduction1 (non-semantic target) in VOT-STB2022.[4]



[4]IV: Illumination Variation, POC: Partial Occlusion, DEF: Deformation, MB: Motion Blur, CM: Camera Motion, ROT: Rotation, BC: Background Clutter, VC: Viewpoint Change, SV: Scale Variation, FOC: Full Occlusion, FM: Fast Motion, OV: Out-of-View, LR: Low Resolution, ARC: Aspect Ration Change