High-Performance Transformer Tracking

Xin Chen*, Graduate Student Member, IEEE, Bin Yan, Jiawen Zhu†, Huchuan Lu‡, Senior Member, IEEE, Xiang Ruan*, and Dong Wang

Abstract—Correlation has a critical role in the tracking field, especially in recent popular Siamese-based trackers. The correlation operation is a simple fusion method that considers the similarity between the template and the search region. However, the correlation operation is a local linear matching process, losing semantic information and easily falling into a local optimum, which may be the bottleneck in designing high-accuracy tracking algorithms. In this work, to determine whether a better feature fusion method exists than correlation, a novel attention-based fusion network, inspired by the transformer, is presented. This network effectively combines the template and search region features using attention mechanisms. Specifically, the proposed method includes an ego-context augment module based on self-attention and a cross-feature augment module based on cross-attention. First, we present a transformer tracking (named TransT) method based on the Siamese-like feature extraction backbone, the designed attention-based fusion mechanism, and the classification and regression heads. Based on the TransT baseline, we also design a segmentation branch to generate the accurate mask. Finally, we propose a stronger version of TransT by extending it with a multi-template scheme and an IoU prediction head, named TransT-M. Experiments show that our TransT and TransT-M methods achieve promising results on seven popular benchmarks. Code and models are available at https://github.com/chensinx-dlut/TransT-M.

Index Terms—Cross-attention, object tracking, self-attention, siamese tracking, transformer.

I. INTRODUCTION

VISUAL object tracking is a fundamental task in computer vision. It aims to estimate the position and scale of a given target in a video sequence. It has a wide range of applications in robot vision, video surveillance, and unmanned driving. The main challenges of tracking are large occlusion, severe deformation, and interference from similar objects. Many efforts have been made in recent years [1], [2]; however, designing a high-accuracy and real-time tracker is still a challenging task.

For most popular trackers (such as SiamFC [3], SiamRPN [4], and ATOM [5]), the correlation operation plays a critical role in integrating the template or target information into the regions of interest (ROIs). However, the correlation operation is a linear matching process. As shown in Fig. 2, correlation calculates the similarity between the template feature and the ROI feature in the form of a sliding window and linear multiplication. This operation leads to semantic information loss, which limits the tracker to capturing the complicated nonlinear interaction between the template and ROIs. Thus, previous models improve the nonlinear representation ability by introducing fashion structures [6], [7], [8], using additional modules [9], [10], [11], and designing effective online updaters [12], [13], [14]. A natural question to ask is: is there any better feature fusion method than correlation?

Recently, transformers [15] have achieved great success in computer vision [16], [17], [18], benefiting from less inductive bias and global information aggregation. The core of the transformer is the attention mechanism. The self-attention mechanism integrates global information, enhances effective information, and suppresses redundant information. The cross-attention mechanism performs the global nonlinear fusion of input information. The unrestricted receptive field gives attention mechanism less inductive bias than convolution and correlation, thus it can easily learn how to perform feature fusion from large data.

In this work, inspired by the core idea of the transformer and attention, we address the bottleneck of the correlation by designing an attention-based feature fusion network and propose a novel transformer tracking algorithm (named TransT). The proposed feature fusion network consists of an ego-context augment (ECA) module based on self-attention and a cross-feature augment module (CFA) based on cross-attention. ECA enhances the features of a single branch. CFA performs feature fusion of two branches, retaining rich semantic information. ECA and CFA establish dependence between long-distance features and aggregate global information. This fusion mechanism effectively integrates the template and ROI features, thereby producing a more semantic feature map than correlation and leading to more accurate tracking results. Fig. 1 provides some representative visual results, illustrating that our TransT method produces insightful attention maps regarding the target and performs better than other competing trackers [8], [12]. Given that the previous Siamese trackers [3], [4], [6], [7] only use the initial template to compare with the search region, they are not robust to appearance change. When the current target appearance is different from the initial template, the discriminability of the tracker decreases. To address this issue, we further extend our TransT baseline with a multi-template scheme and an IoU

---

*Manuscript received 30 December 2021; revised 23 November 2022; accepted 11 December 2022. Date of publication 27 December 2022; date of current version 5 June 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62293542 and 62022021, in part by Joint Fund of Ministry of Education for Equipment Pre-research under Grant 8091B032155, in part by Central Guidance on Local Science and Technology Development Fund of Liaoning Province under Grant 2022JH610010026, and in part by the Science and Technology Innovation Foundation of Dalian under Grant 2020J26GX036. Recommended for acceptance by S. K. Zhou.

(Corresponding author: Dong Wang.)

Xin Chen, Bin Yan, Jiawen Zhu, Huchuan Lu, and Dong Wang are with the School of Information and Communication Engineering, Dalian University of Technology, Dalian, Liaoning 116024, ChinaNingbo Institute, Dalian University of Technology, Ningbo, Zhejiang 315016, China (e-mail: chensinx3131@mail.dlut.edu.cn; lhchuan@dlut.edu.cn; wdice@dlut.edu.cn).

Xiang Ruan is with the Tiwaki Company Ltd., Kusatsu, Shiga 525-8577, Japan (e-mail: xuanxiang@tiwaki.com).

Digital Object Identifier 10.1109/TPAMI.2022.3232535
We propose a novel transformer tracking framework consisting of feature extraction, transformer-like feature fusion, and head prediction. The transformer-like feature fusion combines the template and search region features using attention rather than correlation.

- We develop our feature fusion network based on an ego-context augment module with self-attention and a cross-feature augment module with cross-attention. Compared with correlation-based feature fusion, our attention-based method adaptively focuses on useful information and establishes associations between distant features, making the tracker obtain better classification and regression results.
- We design a multi-template scheme and an IoU prediction head to record appearance changes, which greatly improves the robustness of the tracker. We also design a segmentation branch to generate the accurate mask.
- Numerous experimental results on many benchmarks show that the proposed tracker performs significantly better than the state-of-the-art algorithms, especially on large-scale LaSOT, TrackingNet, GOT-10k, and VOT2021 datasets. In addition, the proposed TransT and TransT-M run at approximately 54 and 42 fps, respectively, on a GPU, which meets the real-time requirement.
- We develop many versions of our tracker using different backbones and feature fusion networks, which obtain a good trade-off between accuracy and speed.

This study builds upon our conference paper [23] and significantly extends it in various aspects. First, we design the segmentation branch for our tracker to make the bounding box prediction and instance segmentation through a unified framework. Second, we further extend our tracker with the multi-template scheme and the IoU prediction design. The multi-template scheme records the temporal information and the appearance change of the target. The IoU prediction head controls the template update. Third, we provide more variant versions of our method with different numbers of feature fusion layers and backbone architectures. The fastest version reaches 70 fps with high performance. Furthermore, we incorporate the challenging tracking dataset VOT for more comprehensive evaluations and add more trackers for comparison.

### II. RELATED WORK

**Visual Object Tracking.** In recent years, Siamese-based methods [3], [4], [6], [7], [8], [19], [24] have been more popular in the tracking field. SiamFC [3], the pioneering work, combined naive feature correlation with a Siamese framework. After that, SiamRPN [4] combines the Siamese network with RPN [25] and conducted feature fusion using depthwise correlation to obtain more precise tracking results. Some further improvements have been made, such as adding additional branches [19], [21], using deeper architectures [6], and exploiting anchor-free architectures [7], [8]. These mainstream tracking architectures can be divided into two parts: a backbone network to extract image features and a correlation-based network to compute the similarity between the template and the search region. Some popular online trackers (e.g., ECO [26], ATOM [5], and DiMP [12]) also heavily rely on the correlation operation. However, two issues have been overlooked. First, the correlation-based network does not make full use of the global context; thus, falling into the local optimum is easy. Second, through correlation, the semantic information is lost to some degree, which may lead to an imprecise prediction. Therefore, in TransT [23], we design a variant transformer structure based on the attention mechanism to replace the correlation-based network for conducting feature fusion, significantly improving the tracking performance.

Besides our work, TMT [27] employs the transformer and combines it with SiameseRPN [4] and DiMP [12] as a feature enhancement module rather than replace the correlation. Stark [28] proposes a transformer tracking framework by concatenating the search region and the template. It also employs the
Transformer and Attention. Transformer [15] was first introduced by Vaswani et al. and applied in machine translation. Briefly, the transformer is an architecture for transforming one sequence into another with the help of attention-based encoders and decoders. The attention mechanism looks at an input sequence and decides at each step which other parts of the sequence are important. Thus, it facilitates capturing the global information from the input sequence. The transformer has replaced recurrent neural networks in many sequential tasks (natural language processing [29], speech processing [30], [31], and computer vision [32]) and gradually extended to handle nonsequential problems [16], [17]. In [16], Carion et al. consider object detection as a set prediction problem and adopt the encoder-decoder architecture in [15] as the detection head. Experiments on COCO [33] demonstrate that the DETR approach achieves comparable results to an optimized Faster R-CNN baseline [25]. Motivated by the success of DETR and the close relationship between detection and tracking (e.g., RPN [25] and SiamRPN [4]), we attempt to introduce the transformer into the tracking field. Unlike DETR, we do not directly follow the encoder-decoder architecture in the original transformer because it does not well match the tracking task. We adopt the core idea of the transformer and exploit the attention mechanism to design the ego-context augment (ECA) and cross-feature augment (CFA) modules. The integration of ECA and CFA focuses on feature fusion between the template and search region rather than extracting information from only one image in [16]. This design philosophy is suitable for visual object tracking.

Several efforts have been made to introduce the attention mechanism and transformer in the tracking field. ACF [34] learns an attention network to do switching among different correlation filters. MLT [9] adopts channelwise attention to provide the matching network with target-specific information. These two works merely borrow the concept of attention to conduct model or feature selection. To improve tracking performance, different attention layers (such as channelwise attention [35], [36], spatial-temporal attention [37], and residual attention [35]) are utilized to enhance the template information within the correlation matching framework. SiamAttn [10] explores self-attention and cross-branch attention to improve the discriminative ability of target features before applying depthwise cross-correlation. CGACD [11] learns attention from the correlation result of the template and search region, and then adopts the learned attention to enhance the search region features for further classification and regression. These works improved tracking accuracy with the attention mechanism, but they still rely heavily on the correlation operation in fusing the template and search region features. Recently, some works [23], [27], [28] have successfully introduced transformers into tracking and alleviate this dilemma. In this work, we extend our transformer-based tracking method TransT [23].

III. TRANSFORMER TRACKING

This section presents the proposed transformer tracking methods TransT and TransT-M. As shown in Fig. 3, our TransT consists of three fundamental components: the backbone network, the feature fusion network, and prediction head networks. The backbone network separately extracts the features of the template and search region, and then adopts the learned attention to enhance the search region features for further classification and regression. These works improved tracking accuracy with the attention mechanism, but they still rely heavily on the correlation operation in fusing the template and search region features. Recently, some works [23], [27], [28] have successfully introduced transformers into tracking and alleviate this dilemma. In this work, we extend our transformer-based tracking method TransT [23].

A. Overall Architecture

Feature Extraction. Similar to Siamese-based trackers [3], [4], the proposed TransT method takes a pair of image patches (i.e., the template patch \( z \in \mathbb{R}^{3 \times H_{to} \times W_{to}} \) and the search region patch \( x \in \mathbb{R}^{3 \times H_{te} \times W_{te}} \) ) as the inputs of the backbone network. The template patch is expanded by twice the side length from the center of the target in the first frame of a video sequence, which includes the appearance information of the target and its local surrounding scene. The search region patch is expanded four times the side length from the center coordinate of the target

\[ x \]

The tracking results are also postprocessed by the window penalty, which will be introduced in Section IV.
in the previous frame, and the search region typically covers the possible moving range of the target. The search region and template are reshaped to squares and processed by the backbone. We use a modified ResNet50 \cite{he2016deep} backbone for feature extraction. Specifically, we remove the last stage of ResNet50 and take the outputs of the fourth stage as final outputs. We also change the convolution stride of the downsampling unit of the fourth stage from 2 to 1 to obtain a larger feature resolution. In addition, we modify the $3 \times 3$ convolution in the fourth stage to the dilation convolution to increase the receptive field. The backbone processes the search region and the template to obtain their feature maps $f_z \in \mathbb{R}^{H_z \times W_z}$, where $H_z, W_z = \frac{H_s}{8}, \frac{W_s}{8}, H_z, W_z = \frac{H_s}{8}, \frac{W_s}{8}$, and $C = 1024$.

**Feature Fusion Network.** We design a feature fusion network to fuse the features $f_z$ and $f_x$. First, a $1 \times 1$ convolution reduces the channel dimension of $f_z$ and $f_x$, obtaining two feature maps with lower dimension, $f_{z0} \in \mathbb{R}^{d \times H_z \times W_z}$ and $f_{x0} \in \mathbb{R}^{d \times H_z \times W_z}$. We employ $d = 256$ in our implementation. Given that the attention-based feature fusion network takes a set of feature vectors as input, we flatten $f_{z0}$ and $f_{x0}$ in the spatial dimension, obtaining $f_{z1} \in \mathbb{R}^{d \times H_z \times W_z}$ and $f_{x1} \in \mathbb{R}^{d \times H_z \times W_z}$. $f_{z1}$ and $f_{x1}$ can be regarded as a set of feature vectors of length $d$. As shown in Fig. 3, the feature fusion network takes $f_{z1}$ and $f_{x1}$ as the inputs to the template branch and the search region branch, respectively. First, two ego-context augment (ECA) modules enhance the useful semantic context adaptively by multi-head self-attention. Then, two cross-feature augment (CFA) modules receive the feature maps of two branches and fuse them through multi-head cross-attention. In this way, two ECAs and two CFAs form a fusion layer, as shown in the dotted box in Fig. 3. The fusion layer repeats $N$ times, followed by an additional CFA to fuse the feature map of two branches, decoding a feature map $f \in \mathbb{R}^{d \times H_z \times W_z}$. We employ $N = 4$ by default and also provide other versions with different $N$. The details of the ECA and CFA modules are introduced in Section III-B.

**Prediction Head Network.** The prediction head consists of a classification branch and a regression branch, where each branch is a three-layer perceptron with hidden dimension $d$ and a ReLU activation function. For the feature map $f \in \mathbb{R}^{d \times H_z \times W_z}$ generated by the feature fusion network, the head makes predictions on each vector to obtain $H_zW_z$ foreground/background classification results and $H_zW_z$ normalized coordinates with respect to the search region size. Our tracker directly predicts the normalized coordinates instead of adjusting anchor points or anchor boxes, thereby making the tracking framework more concise.

**B. Ego-Context Augment and Cross-Feature Augment Modules**

**Multi-Head Attention.** Attention is the fundamental component in designing our feature fusion network. Given queries $Q$, keys $K$ and values $V$, the attention function is defined in \cite{vaswani2017attention}.

\[
\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^\top}{\sqrt{d_k}} \right) V, \tag{1}
\]

where queries $Q$ and keys $K$ are two sets of feature vectors with dimension $d_k$, and values $V$ are a set of feature vectors with dimension $d_v$. The attention score between $Q$ and $K$ is obtained by the scaled dot-product operation, and then the attention map is generated by the $\text{softmax}$ operation. The algorithm reweights $V$ according to the attention map. In this way, according to the association between $Q$ and $K$, the attention mechanism adaptively focuses on the important positions in $V$.

As described in \cite{wang2019video}, extending the attention mechanism \cite{vaswani2017attention} into multiple heads enable the mechanism to consider various attention distributions and make the model focus on different information aspects. The mechanism of multi-head attention is defined in \cite{wang2019video}. We refer the reader to the literature \cite{wang2019video} for more detailed descriptions.

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, \ldots, H_n)W^Q, \tag{2}
\]

\[
H_i = \text{Attention}(QW^{Q}_i, KW^{K}_i, VW^{V}_i), \tag{3}
\]
receives the $P$, and $W$ are parameter matrices. In the present work, $R \in \mathbb{R}$ and $W = b d W$ is the spatial positional encoding and $R \in X = 256 \in \mathbb{R}$ and $W + W R + b$, $b W \in \mathbb{R}$, is the output of ECA.

$\mathbf{P} \in \mathbb{R}^{d \times d}$ receives the feature from $P + \text{MultiHead}(W P d + = b d W)$ and then reweighs $X$ on the left side of Fig. 4. Ego-Context Augment (ECA). The structure of ECA is shown on the left side of Fig. 4. ECA adaptively integrates the information from different positions of the feature map by using multi-head self-attention in the residual form. As shown in (1), the attention mechanism has no ability to distinguish the position information of the input feature sequence. Thus, we implement a spatial positional encoding to the input $X \in \mathbb{R}^{d \times N}$. Following [16], we use a sine function to generate spatial positional encoding. Finally, the mechanism of ECA can be summarized as

$$X_{EC} = X + \text{MultiHead}(X + P_x, X + P_x, X), \quad (4)$$

where $P_x \in \mathbb{R}^{d \times N}$ is the spatial positional encoding and $X_{EC} \in \mathbb{R}^{d \times N}$ is the output of ECA.

Cross-Feature Augment (CFA). The structure of CFA is shown in the right of Fig. 4. CFA fuses the feature vectors from two inputs by using multi-head cross-attention in the form of residuals. Similar to ECA, spatial positional encoding is also used in CFA. Furthermore, an FFN module is used to enhance the fitting ability of the model, which is a fully connected feed-forward network that consists of two linear transformations with a ReLU in between. That is,

$$\text{FFN} (x) = \max (0, xW_1 + b_1) W_2 + b_2, \quad (5)$$

where $W$ and $b$ represent weight matrices and basis vectors, respectively. The subscripts denote different layers.

Thus, the mechanism of CFA can be summarized as follows:

$$X_{CF} = \bar{X}_{CF} + \text{FFN}\left(\bar{X}_{CF}\right),$$

$$\bar{X}_{CF} = X_q + \text{MultiHead}(X_q + P_q, X_{kv} + P_{kv}, X_{kv}), \quad (6)$$

where $X_q \in \mathbb{R}^{d \times N}$ is the input of the branch that the module is applied, $X_{kv} \in \mathbb{R}^{d \times N_{kv}}$ is the input from another branch, $P_q \in \mathbb{R}^{d \times N_q}$ is the spatial positional encoding corresponding to $X_q$, $P_{kv} \in \mathbb{R}^{d \times N_{kv}}$ is the spatial positional encoding corresponding to $X_{kv}$, and $X_{CF} \in \mathbb{R}^{d \times N}$ is the output of CFA. According to (6), CFA calculates the attention map according to multiple scaled products between $X_{kv}$ and $X_q$ and then reweights $X_{kv}$ according to the attention map, and adds it to $X_q$ to fuse features of two branches.

C. Discussion

Differences From the Original Transformer. The right of Fig. 5 shows the detailed architecture of the feature fusion network of our TransT, which is a modified version of the transformer. We also design a tracker which employs the original transformer, named TransT(ori). The left of Fig. 5 shows the detailed architecture of the original transformer that we employ in TransT(ori). In TransT(ori), the transformer takes template vectors as the encoder’s input and search region vectors as the decoder’s input rather than the opposite, because we found that the performance is better in this way. We provide more discussion about this issue in Section IV-D. The original transformer is unsuitable for the tracking framework. First, the transformer is asymmetric, and the processing of the two inputs is inconsistent. Cross-attention only reweights the template vectors but not the search vectors, which is unreasonable. Second, self-attention is used more than cross-attention, which does not match the tracking task well because the interaction between the template and search region is much more important in tracking. For these reasons, our method draws on the core idea of the transformer,

![Fig. 4. Left: ECA module. Right: CFA module. The ECA module is based on multi-head self-attention in a residual form. The CFA module is based on multi-head cross-attention and FFN in a residual form. The input $X_q$ receives the feature from the branch where CFA is located, and $X_{EC}$ receives the feature from the other branch. Spatial positional encodings are used to implement position information. ECA enhances the contextual information of the inputs, and CFA adaptively fuses the features from two branches.]

![Fig. 5. Detailed architecture of our feature fusion network and the original transformer. Left: original transformer architecture used in TransT(ori). Right: feature fusion network of TransT. The parameters of TransT and TransT(ori) are 23.0 and 26.4M, respectively.]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
IV-D shows the detailed architecture of the segmentation and concatenate it with the two attention maps. At this point, the inputs of $n=1$ to fuse the concatenated feature with the $n \times n \times (7 \times 7)$, the template gives no information, and the attention module attempts to see all objects that are different from the background. The same thing happens in the second line, that is, the self-attention map of the template. The third and fourth attention maps for the template no longer follow the initial spatial position but show a puzzling distribution. The underlying reason is that after the target is identified, the features of the template branch no longer need to keep the information of the template but store much of the target’s boundary information, becoming a feature library serving for regression. Through visualizing the attention maps, one can see that the attention modules automatically look for global useful information, thereby making the tracker achieve good results.

D. Segmentation Branch

Fig. 7 shows the detailed architecture of the segmentation branch. The template vectors are the output feature vectors of the last feature fusion layer’s template branch. The fusion vectors are the output feature vectors of the feature fusion network. In the template vectors, we get the feature vector corresponding to the center position of the template patch (indicated as the center vector in Fig. 7) and employ an additional multi-head attention to calculate an attention map from it to the fusion vectors. We employ a 8-head attention layer to obtain this attention map with the shape of $32 \times 32 \times 8$. In the fusion vectors, we get the vector with the highest confidence score predicted by the classification head and employ another 8-head attention layer to calculate an attention map from it to the fusion vectors. Then, we obtain another attention map with the shape of $32 \times 32 \times 8$. After that, we reshape the fusion vectors back in the spatial dimension to $32 \times 32 \times 256$ and concatenate it with the two attention maps. We employ FPN [39] to fuse the concatenated feature with the low-level feature pyramid of the search region (from stages 1 to 4 of the backbone network). The shape of the output feature of the FPN is $128 \times 128 \times 8$. Finally, we use a mask prediction head to predict the segmentation mask with the shape of $256 \times 256 \times 1$. The mask prediction head is a $3 \times 3$ convolutional layer followed by a bilinear interpolation.

E. TransT-M

As shown in Fig. 3, we extend TransT with a multi-template scheme and an IoU prediction head to produce a stronger tracker named TransT-M. The multi-template scheme records historical appearances of the target, making the tracker more robust to the appearance change. The IoU prediction head controls the

---

Fig. 6. Visualization of the attention maps for a representative image pair. From left to right, the feature fusion layer goes deeper. From top to bottom, they are self-attention maps in the search region, self-attention maps in the template, cross-attention maps in the search region, and cross-attention maps in the template.
template update by comparing the predicted IoU with the update threshold.

**Multi-Template Scheme.** In the tracking process, sometimes the appearance of the target changes greatly. Finding the target only by relying on the template of the first frame, especially in the case of distractors existing, is difficult for the tracker. Thus, we use multiple templates to record the target’s historical appearance. Specifically, we use $M$ templates in online tracking. In the $M$ templates, we always maintain the initial template as it is precise and reliable. For the rest of the $M - 1$ templates, we update them with the template collected during tracking; these templates are not as precise as the initial templates, but they provide more information about the appearance of the target. We use the backbone to extract the features from these templates. Then, after reshaping them to 2D features, we concatenate them in the spatial dimension and input the concatenated feature to the template branch of the feature fusion network. We employ $M = 2$ by default.

In our design, instead of having different designs for initial and updated templates, we treat them equally. This may not be the best choice to pursue higher performance. Empirically, designing different branches for different templates may give better results. However, we find that the simple method to treat templates as equal works well. To keep the framework concise, we do not employ extra designs for different templates. Another advantage of this choice is that the model can take different numbers of templates as input without redesigning or retraining, making it flexible in application. The users can freely select the single-template setup for faster speed or multi-template setup for better performance. This characteristic is also due to the following reasons. First, attention can receive different numbers of feature vectors as inputs. The results of the attention layer are calculated by the $softmax$ weighted average using the attention score. For example, the result of inputting one template is the same as that of inputting multiple copies of this template. Second, we use the concatenating method to combine the multi-template features. In the concatenation method, different feature vectors are independent, thereby ensuring the attention layers have seen independent initial-template feature vectors and updated-template feature vectors in training. Thus, the model learned how to deal with these two types of feature vectors. In testing, even if the number of templates is different, each template feature vector is no more than one of these two types. Third, under different training and testing numbers of templates, the proportions of the initial-template feature vectors are different, which creates some training-testing inconsistency. However, in the experiments, we find that the impact of such inconsistency is negligible. One underlying reason is that the updated templates in training are generated by random jitter, which introduces much diversity in template vectors. In Section IV, we provide the results of the TransT-M training and testing with different template numbers.

**IoU Prediction Head.** The tracker needs an indicator to control the update of the template. However, the confidence score is not a good indicator, because it represents the possibility of the target existing at each feature vector’s corresponding location but not indicate whether this feature vector’s predicted bounding box is suitable for being a template. Thus, we design an IoU prediction head to determine whether the prediction can be used as a template. The IoU prediction head is simple. We concatenate the regression vectors and the fusion vectors in the channel dimension and use a three-layer perceptron to predict the IoU on each vector. Then, we compare the predicted IoU with a update threshold to control the update of the templates.

**F. Training Loss**

**Classification and Regression Prediction Loss.** The prediction head receives $H_x \times W_x$ feature vectors and outputs $H_x \times W_x$ binary classification and regression results. We select the prediction of feature vectors corresponding to pixels in the ground-truth bounding box as positive samples, and the rest are negative samples. All samples contribute to the classification loss, whereas only positive samples contribute to the regression loss. To reduce the imbalance between positive and negative samples, we downweighted the loss produced by negative samples by a
are the regularization parameters in our experiments.

\[ L_{cls} = - \sum_j [y_j \log(p_j) + (1 - y_j) \log(1 - p_j)], \tag{7} \]

where \( y_j \) denotes the ground-truth label of the \( j \)-th sample, \( y_j = 1 \) denotes the foreground, and \( p_j \) denotes the probability of the sample belonging to the foreground predicted by the learned model. For regression, we employ a linear combination of \( l_1 \)-norm loss \( L_1(\cdot, \cdot) \) and the generalized IoU loss \( L_{GIoU}(\cdot, \cdot) \) \[40]. The regression loss can be formulated as

\[ L_{reg} = \sum_j \hat{y}_{[y_j=1]} [\lambda_G L_{GIoU}(b_j, \hat{b}) + \lambda_1 L_1(b_j, \hat{b})], \tag{8} \]

where \( y_j = 1 \) denotes the positive sample, \( b_j \) denotes the \( j \)-th predicted bounding box, and \( \hat{b} \) denotes the normalized ground-truth bounding box. \( \lambda_G = 2 \) and \( \lambda_1 = 5 \) are the regularization parameters in our experiments.

**IoU Prediction Loss.** The IoU prediction head outputs \( H \times W \) IoUs. We calculate each positive sample’s IoU with the corresponding ground-truth bounding box as the IoU ground truth. We use the MSE loss for the IoU prediction head. Only the positive samples contribute to this loss because we only need the IoU prediction to judge the quality of the bounding box rather than to judge the positive and negative samples.

**Segmentation Prediction Loss.** For the segmentation branch, we use the dice loss \[41\] and the focal loss \[42\].

\[ L_{seg} = \lambda_D L_{Dice}(m, \hat{m}) + \lambda_f L_{focal}(m, \hat{m}), \tag{9} \]

where \( m \) is the predicted mask and \( \hat{m} \) is the ground truth. \( \lambda_D = 1 \) and \( \lambda_f = 1 \) are the regularization parameters in our experiments.

**A. Implementation Details**

**Offline Training.** For TransT, we train our model on the training splits of COCO \[33\], TrackingNet \[44\], LaSOT \[43\], and GOT-10k \[45\] datasets. For the video datasets (TrackingNet, LaSOT, and GOT-10k), we directly sample the image pairs from one video sequence to collect training samples. For COCO dataset, we apply some transformations to the original image to generate image pairs. Common data augmentation (such as translation and brightness jitter) is applied to enhance the training set. The sizes of the search region patch and the template patch are \( 256 \times 256 \) and \( 128 \times 128 \), respectively. The backbone parameters are initialized with ImageNet-pretrained ResNet-50 \[38, 63\], and other parameters of our model are initialized with Xavier initialization \[64\]. We train the model with AdamW \[65\], setting the backbone’s learning rate to 1e-5, other parameters’ learning rate to 1e-4, and weight decay to 1e-4. We train the network on eight NVIDIA Titan RTX GPUs with 14 sample pairs on each GPU (batch size 112) for 500 epochs with 80,000 sample pairs per epoch. The learning rate decreases by a factor of 10 after 400 epochs.

For TransT-M, we divide the training process into two stages. First, we train the multi-template TransT baseline. We use \( M = 2 \), that is, two templates by default. For the video datasets, we sample \( M + 1 \) frames in a video sequence to form a sample unit. To simulate the online tracking process, we select the first frame or the last frame of a sample unit to generate the search region. We use the frame farthest from the search region frame to generate the initial-template patch and use the middle frames to generate the updated-template patches. For the COCO dataset, we apply transformations to a image to generate a sample unit. In a sample unit, we randomly select one image to get the search region patch, another image to get the initial-template patch, and the rest to get the updated-template patches. For the search region patch and the updated templates, we jitter and transform the position and shape of the target box when we obtain the patches to simulate the inaccuracy of template updating in tracking. For the initial template, we do not apply jittering or transformation because the initial target’s bounding box given in online tracking is generally accurate. The other training settings are the same as those in TransT. For the second stage, we freeze the parameters of the multi-template TransT baseline and train the IoU prediction head. For the IoU prediction head, we use the same training datasets as the TransT baseline. We employ AdamW with a learning rate of 1e-3 and weight decay of 1e-4. We train the IoU prediction head on two NVIDIA Titan RTX GPUs with a batch size of 256 for 90 epochs with 200 iterations per epoch. The learning rate decreases by a factor of 10 after every 30 epochs. We do not train the IoU prediction head with the TransT baseline together because we hope that the ground truth of the IoU prediction head is consistent with the final network’s bounding box prediction and unaffected by the bad bounding box prediction at the initial stage of training. For the segmentation branch, we train it using YouTube-VOS \[66\] and some saliency datasets \[67, 68, 69\], which is consistent with Alpha-Refine \[21\]. The training strategy is the same as the IoU prediction head, except that the initial learning rate is set to 1e-2. We do not train the segmentation branch with the TransT baseline together, because the segmentation datasets are relatively small and we do not want it to be flooded by the tracking datasets.

**Online Tracking.** In online tracking, the prediction head outputs 1024 boxes with their confidence scores. We reshape the score map to the shape of \( 32 \times 32 \). Then, the window penalty is adopted for postprocessing these scores. Specifically, the Hanning window with the shape of \( 32 \times 32 \) is applied to the score map, weighted by a parameter \( w \) (chosen as 0.49 by default). The final score \( score_w \) can be defined as

\[ score_w = (1 - w) \times score + w \times score_h, \tag{10} \]

where \( score \) is the original score map of the tracker’s output. \( score_h \) is the value of the corresponding position on the Hanning window, and \( score_w \) is the final score map. Through window penalty, the confidence scores of feature points far from the target in the previous frames are punished. Finally, we select the box with the highest score as the tracking result. For TransT, we always use the initial target’s bounding box of the first frame as the template and do not update it. For TransT-M, we use two templates by default, and we always keep the initial template and update other templates. The update of the templates is controlled by the predicted IoU. If the predicted IoU is higher than the
update threshold (chosen as 0.89 by default), the template is updated using the predicted result.

B. Evaluation on TrackingNet, LaSOT and GOT-10k

In this subsection, we compare our method with state-of-the-art trackers on the large-scale LaSOT [43], TrackingNet [44], and GOT-10k [45] datasets. The results are shown in Table I.

LaSOT. LaSOT [43] is a recent large-scale dataset, which contains 1,400 challenging videos: 1,120 for training and 280 for testing. We follow the one-pass evaluation (Success and Precision) to compare different tracking algorithms on the LaSOT test set. Then, we report the Success (AUC) and Precision (P and PNorms) scores in Table I. The proposed TransT-M and TransT achieve competitive performance. Compared with two transformer-based tackers TrDiMP and TrSiam, our TransT-M method performs 1.5% and 3.0% higher, respectively, in AUC. Fig. 8 shows an attribute-based evaluation of representative state-of-the-art algorithms, illustrating that our trackers achieve competitive results on many challenging attributes.

TrackingNet. TrackingNet [44] is a large-scale tracking dataset that covers diverse object classes and scenes. Its test set contains 511 sequences of publicly available ground truth. We submit our tracker’s outputs to the official online evaluation server and report the Success (AUC) and Precision (P and PNorms) results in Table I. Our TransT obtains 81.4%,

| Method      | Year | LaSOT [43] AUC | LaSOT [43] P | LaSOT [43] PNorm | TrackingNet [44] AUC | TrackingNet [44] P | TrackingNet [44] PNorm | GOT-10k [45] AO | GOT-10k [45] SR0.5 | GOT-10k [45] SR0.75 |
|-------------|------|----------------|--------------|------------------|-----------------------|----------------------|-----------------------|----------------|----------------|------------------|
| TransT-M    | ours | 65.4           | 73.9         | 69.6             | 82.5                  | 87.3                  | 80.0                  | 74.7           | 85.5           | 71.3             |
| TransT      | ours | 64.9           | 73.8         | 69.0             | 81.4                  | 86.7                  | 80.3                  | 72.3           | 82.4           | 68.2             |
| TransT-GOT  | ours | -              | -            | -                | -                     | -                    | -                     | 67.1           | 76.8           | 60.9             |
| ToMP50 [46] | 2022 | 67.6           | -            | -                | 81.2                  | 86.4                  | 78.9                  | -              | -              | -                |
| CSWinTT [47]| 2022 | 66.2           | 75.2         | 70.9             | 81.9                  | 86.7                  | 79.5                  | 69.4           | 78.9           | 65.4             |
| UTT [48]    | 2022 | 64.6           | 67.2         | -                | 79.7                  | -                     | -                     | 67.2           | 76.3           | 60.5             |
| RBO [49]    | 2022 | 55.8           | 57.0         | -                | -                     | -                     | -                     | 64.4           | 76.7           | 50.9             |
| HybTransT [50]| 2022   | 64.5           | 73.9         | -                | -                     | -                     | -                     | -              | -              | -                |
| TransInMo [51]| 2022   | 65.3           | 74.6         | 69.9             | 81.6                  | -                     | -                     | -              | -              | -                |
| ARDiMPsuper [21]| 2021 | 65.3           | 73.2         | 68.0             | 80.5                  | 85.6                  | 78.3                  | 70.1           | 80.0           | 64.2             |
| TrDiMP [27] | 2021 | 63.9           | -            | 61.4             | 78.4                  | 83.3                  | 73.1                  | 68.8           | 80.5           | 59.7             |
| TrSiam [27] | 2021 | 62.4           | -            | 60.0             | 78.1                  | 82.9                  | 72.7                  | 67.3           | 78.7           | 58.6             |
| STMTtrack [52]| 2021 | 60.6           | 69.3         | 63.0             | 80.3                  | 85.1                  | 76.7                  | 64.2           | 73.7           | 57.5             |
| SiamBAN-ACM [53]| 2021 | 57.2           | -            | -                | 75.3                  | 81.0                  | 71.2                  | -              | -              | -                |
| SiamGAT [54] | 2021 | 53.9           | 63.3         | 53.0             | -                     | -                     | -                     | 62.7           | 74.3           | 48.8             |
| DStTrp [55] | 2021 | 43.4           | 51.3         | -                | 64.9                  | 58.9                  | -                     | -              | -              | -                |
| SiamR-CNN [56]| 2020 | 64.8           | 72.2         | -                | 81.2                  | 85.4                  | 80.0                  | 64.9           | 72.8           | 59.7             |
| Ocean [8]   | 2020 | 56.0           | 65.1         | 56.6             | -                     | -                     | -                     | 61.1           | 72.1           | 47.3             |
| KYS [57]    | 2020 | 55.4           | 63.3         | -                | 74.0                  | 80.0                  | 68.8                  | 63.6           | 75.1           | 51.5             |
| DCFST [58]  | 2020 | -              | -            | -                | 75.2                  | 80.9                  | 70.0                  | 63.8           | 75.3           | 49.8             |
| SiamFC++ [7]| 2020 | 54.4           | 62.3         | 54.7             | 75.4                  | 80.0                  | 70.5                  | 59.5           | 69.5           | 47.9             |
| PrDiMP [14] | 2020 | 59.8           | 68.8         | 60.8             | 75.8                  | 81.6                  | 70.4                  | 63.4           | 73.8           | 54.3             |
| CGACD [11]  | 2020 | 51.8           | 62.6         | -                | 71.1                  | 80.0                  | 69.3                  | -              | -              | -                |
| SiamAttn [10]| 2020 | 56.0           | 64.8         | -                | 75.2                  | 81.7                  | -                     | -              | -              | -                |
| MAML [59]   | 2020 | 52.3           | -            | -                | 75.7                  | 82.2                  | 72.5                  | -              | -              | -                |
| DS3 [20]    | 2020 | -              | -            | -                | 72.8                  | 76.8                  | 66.4                  | 59.7           | 67.6           | 46.2             |
| SiamCAR [60] | 2020 | 50.7           | 60.0         | 51.0             | -                     | -                     | -                     | 56.9           | 67.0           | 41.5             |
| SiamBAN [61]| 2020 | 51.4           | 59.8         | 52.1             | -                     | -                     | -                     | -              | -              | -                |
| DiMP [12]   | 2019 | 56.9           | 65.0         | 56.7             | 74.0                  | 80.1                  | 68.7                  | 61.1           | 71.7           | 49.2             |
| SiamPRN++ [6]| 2019 | 49.6           | 56.9         | 49.1             | 73.3                  | 80.0                  | 69.4                  | 51.7           | 61.6           | 32.5             |
| ATOM [5]    | 2019 | 51.5           | 57.6         | 50.5             | 70.3                  | 77.1                  | 64.8                  | 55.6           | 63.4           | 40.2             |
| ECO [26]    | 2017 | 32.4           | 33.8         | 30.1             | 55.4                  | 61.8                  | 49.2                  | 31.6           | 30.9           | 11.1             |
| MDNet [62]  | 2016 | 39.7           | 46.0         | 37.3             | 60.6                  | 70.5                  | 56.5                  | 29.9           | 30.3           | 9.9              |
| SiamFC [3]  | 2016 | 33.6           | 42.0         | 33.9             | 57.1                  | 66.3                  | 33.3                  | 34.8           | 35.3           | 9.8              |

The best three results are shown in red, blue, and green fonts.

![Fig. 8. AUC scores of different attributes on the LaSOT dataset.](image-url)
86.7%, and 80.3% in terms of AUC, P\textsubscript{Norms}, and P, respectively. TransT-M further improves the performance to 82.5%, 87.3%, and 80.0% in AUC, P\textsubscript{Norms}, and P, respectively. TransT-M achieves the best performance. Compared with the second-best tracker CSWinTT [47], TransT-M performs 0.6% higher in AUC. Compared with two transformer-based trackers TrDiMP and TrSiam, TransT-M performs 4.1% and 4.4% better in AUC.

**GOT-10k.** The GOT-10k [45] dataset contains 10k sequences for training and 180 for testing. We follow the defined protocol presented in [45] and submit the tracking outputs to the official evaluation server. Then, we report the obtained results (i.e., AO and SR\textsubscript{T}) in Table I. TransT-GOT denotes training with only the GOT-10k training set. Our methods achieve the best performance. TransT-M and TransT obtain 74.7% and 72.3% AO, respectively, which is much better than previous methods. Compared with the third-best tracker ARDiMP\textsubscript{super} [21], TransT-M and TransT perform 4.6% and 2.2% better in AO.

**C. Evaluation on VOT**

We evaluate our tracker on Visual Object Tracking Challenge 2021 (VOT2021) [22] using the baseline and real-time experiments. VOT2021 consists of 60 challenging videos with mask annotation. VOT2021 adopts expected average overlap (EAO) as the main metric, which simultaneously considers the trackers’ accuracy and robustness. The tracker can submit the mask predictions or bounding box predictions, but the EAO calculation uses the mask annotation. We use the official evaluation tool and submit the mask prediction. Fig. 9 shows the EAO plots of the proposed methods and other trackers from the official VOT challenge. TransT-SEG is the TransT baseline with our segmentation branch. The results show that our tracker TransT-SEG achieves competitive results in the real-time experiment. Furthermore, TransT-M achieves the best result in the real-time experiment and the third-best result in the baseline experiment. Thus, it becomes the winner of VOT-RT2021. In the real-time experiment, TransT-M performs 1.9% better than the second-best tracker STARK [28], which is also a transformer-based tracker. There are many appearance changes and similar target interference in VOT’s video sequences. Thus, TransT-M performs much better than TransT-SEG on VOT, due to the strong discriminative ability.

**D. Ablation Study and Analysis**

*Postprocessing:* In prior works, such as SiamRPN [4], SiamRPN++ [6] and Ocean [8], the final tracking results are selected by postprocessing schemes, including cosine window penalty, scale change penalty, and bounding box smoothing. However, these postprocessing schemes are parameter-sensitive because three hyperparameters need to be adjusted carefully for different test sets. To alleviate this problem, in this work, we merely adopt the window penalty to conduct postprocessing. To show the effect of this simple postprocessing, we compare the TransT variants with and without the postprocessing step in Table II. TransT denotes our tracker, and TransT-np is our tracker without postprocessing. First, from Table II, we can conclude that our TransT without postprocessing still achieves state-of-the-art performance, which is attributed to the transformer-like fusion method. Second, the postprocessing step further improves the tracking accuracy, producing the best record among almost all metrics on these benchmarks.

*Comparison With the Original Transformer.* To show the superiority of our feature fusion network, we design a tracker using the original transformer. As shown in Fig. 5 and introduced in Section III-C, we replace the feature fusion network with the original transformer structure and keep the other components unchanged. We input the template feature to the encoder and the search region feature to the decoder. The comparison results are shown in Table II. TransT(ori) denotes the tracker with the original transformer, and TransT(ori)-np is the TransT(ori) method without postprocessing. First, the TransT(ori)-np variant achieves an AUC score of 60.9% on LaSOT, an AUC score of 80.9% on TrackingNet, and an AO score of 68.6% on GOT-10k, which are also better than many state-of-the-art correlation-based methods. This indicates that the transformer structure works better than the simple correlation operation in dealing with feature fusion. Second, by observing TransT v.s. TransT(ori) and TransT-np v.s. TransT(ori)-np, we can conclude that the proposed transformer performs better than the original transformer structure by a large margin. In addition, we also see that the postprocessing works for both the TransT and TransT(ori) methods.

We also evaluate inputting the search region feature to the encoder and the template feature to the decoder, which is denoted as TransT(ori)-opp. In TransT(ori)-opp, the tracker predicts results on each template feature vector; thus, we cannot divide the positive and negative samples according to the ground-truth box and cannot apply the postprocessing. We set every template feature vector’s prediction as a positive sample. The results show that TransT(ori)-opp is not as good as TransT(ori) for two reasons. First, the ground-truth matching independent of the location of the search region leads to all feature vectors tending to predict the same target, which lacks diversity. All feature vectors can easily give the same wrong prediction. In contrast,
the correct prediction can be selected from the diverse prediction through confidence score and postprocessing in TransT(ori). Second, this setting lacks template feature enhancement through cross-attention, which is very important. We exchange the input of the search region feature and the template feature of TransT, which is denoted as TransT-opp. In this setting, we find that the performance degradation is not as much as TransT(ori)-opp because it keeps the enhancement of the template feature through cross-attention.

Comparison With Correlation. Prior Siamese trackers use the cross-correlation operation to compute the similarity between the template feature and search region feature. Fig. 2(a) illustrates the cross-correlation layer. The template feature is used as a convolution kernel to convolve with the search region feature to generate a single-channel response map. The response map can measure the similarity between the template and search region. SiamRPN++ proposed a depthwise cross-correlation layer, as shown in Fig. 2(b), and two feature maps with the same number of channels perform the correlation operation channel by channel to generate a multi-channel response map. The correlation operation plays a critical role in the tracking field. However, correlation is a linear calculation without parameters. This single feature fusion process leads to semantic loss, which weakens its feature fusion ability. Compared with correlation-based methods, first, our attention-based method can establish long-distance feature associations, which effectively aggregate the global information of the template and search region. Second, our method outputs features with rich semantic information, not just a similarity map. We conduct experiments to compare CFA with correlation and explore the impact of ECA. To make a fair comparison, for the TransT without CFA we keep the FFN in CFA unchanged, remove only the cross-attention layers, and replace the last CFA module with depthwise correlation. The comparison results are shown in Table III. The comparison results show that after replacing CFA with the correlation layer, the performance significantly decreases. Without ECA, the performance of the tracker drops. Without ECA and CFA, the performance further drops, and the impact of the postprocessing becomes greater. These results show that without attention modules, the localization ability of the tracker significantly decreases, and it needs to rely more on the prior information in postprocessing.

Different $N$. We explore the effect of $N$, i.e., the number of the feature fusion layers. The results are shown in Table IV. Param indicates the number of parameters, Model Time represents the time of one forward propagation of the model, and Speed represents the actual running speed of the algorithm. For $N = 2, 4, 6$, the number of parameters are 16.7, 23.0, and 29.3M, respectively, and their speeds are 71, 54, and 42 fps, respectively, on an NVIDIA Titan RTX GPU. We found that $N = 4$ is better than $N = 2$, but $N = 6$ is slightly worse than $N = 4$ in general. In training, the final loss value of $N = 6$ is lower than $N = 4$, and the lower performance on test sets is due to overfitting. The same phenomenon occurs when we use a larger backbone. Thus, we choose $N = 4$ as the default setting to balance model size, speed, and performance. The $N = 2$ version also achieves promising results with a higher speed of 71 fps.

Different Backbones. The results of employing different backbones are shown in Table V. The results show that the tracker with ResNet18 also achieves good performance with fewer parameters and faster speed. However, the performance of the tracker with ResNet101 drops slightly because of overfitting. For ResNet18, ResNet50, and ResNet101, the number of parameters are 17.1, 23.0, and 42.0M, respectively, and their speeds are 61,

---

**Table II**

| Method          | LaSOT [43] | TrackingNet [44] | GOT-10k [45] |
|-----------------|------------|------------------|---------------|
|                 | AUC        | $P_{N_{norm}}$   | $P_{N_{norm}}$ | AO | SR@0.5 | SR@0.75 |
| TransT          | 64.9       | 73.8             | 69.0          | 81.4 | 86.7   | 80.3     | 72.3 | 82.4   | 68.2     |
| TransT-opp      | 62.9       | 71.5             | 66.9          | 81.1 | 86.4   | 80.0     | 71.5 | 81.5   | 67.5     |
| TransT-opp      | 63.5       | 72.9             | 67.4          | 77.8 | 85.5   | 80.9     | 69.9 | 80.1   | 65.0     |
| TransT(ori)     | 62.3       | 71.1             | 66.2          | 81.3 | 86.1   | 78.9     | 70.3 | 80.2   | 65.8     |
| TransT(ori)-opp | 60.9       | 69.4             | 64.8          | 80.9 | 85.6   | 78.4     | 68.6 | 78.2   | 65.1     |
| TransT(ori)-opp | 56.9       | 64.6             | 58.4          | 60.6 | 67.7   | 64.0     | 61.9 | 70.4   | 55.1     |

*The best results are shown in the red font.*

**Table III**

| Method  | ECA | CFA | Correlation | AUC | $P_{N_{norm}}$ | $P_{N_{norm}}$ | AO | SR@0.5 | SR@0.75 |
|---------|-----|-----|-------------|-----|---------------|---------------|----|--------|--------|
| TransT  | √   | √   |             | 64.9| 73.8          | 69.0          | 81.4| 86.7   | 80.3   |
| TransT  | √   |     |             | 62.9| 71.9          | 66.2          | 81.1| 86.2   | 79.1   |
| TransT  |     | √   |             | 57.7| 65.4          | 59.5          | 77.5| 82.2   | 74.0   |
| TransT  |     |     |             | 47.7| 48.6          | 41.7          | 68.8| 71.4   | 60.9   |
| TransT-opp | √  | √   |             | 62.9| 71.5          | 66.9          | 81.1| 86.4   | 80.0   |
| TransT-opp |     | √   |             | 61.0| 69.6          | 64.5          | 80.0| 85.0   | 77.9   |
| TransT-opp |     |     |             | 57.3| 65.2          | 58.8          | 76.2| 80.8   | 72.8   |
| TransT-opp |     |     |             | 35.3| 17.9          | 20.1          | 46.5| 40.3   | 27.4   |

*The best results are shown in the red font.*

---

**Table IV**

| Method   | ECA | CFA | Correlation | AUC | $P_{N_{norm}}$ | $P_{N_{norm}}$ | AO | SR@0.5 | SR@0.75 |
|----------|-----|-----|-------------|-----|---------------|---------------|----|--------|--------|
| TransT   | √   | √   |             | 64.9| 73.8          | 69.0          | 81.4| 86.7   | 80.3   |
| TransT   | √   |     |             | 62.9| 71.9          | 66.2          | 81.1| 86.2   | 79.1   |
| TransT   |     | √   |             | 57.7| 65.4          | 59.5          | 77.5| 82.2   | 74.0   |
| TransT   |     |     |             | 47.7| 48.6          | 41.7          | 68.8| 71.4   | 60.9   |
| TransT-opp | √  | √   |             | 62.9| 71.5          | 66.9          | 81.1| 86.4   | 80.0   |
| TransT-opp |     | √   |             | 61.0| 69.6          | 64.5          | 80.0| 85.0   | 77.9   |
| TransT-opp |     |     |             | 57.3| 65.2          | 58.8          | 76.2| 80.8   | 72.8   |
| TransT-opp |     |     |             | 35.3| 17.9          | 20.1          | 46.5| 40.3   | 27.4   |

*The best results are shown in the red font.*
We employ ResNet50 as the default backbone to balance model size, speed, and performance. The ResNet18 version also obtains good results with a higher speed of 61 fps. Different M. We train TranT-M with two templates. However, TranT-M can take different numbers of templates as input without retraining and affecting the performance. The results of different M, that is, the number of templates, are shown in Tables VI and VII. For the long-term LaSOT dataset, we do not update the templates. We also provide the results of TranT-M using different N and backbone networks in these two tables.

When we use different M, we do not retrain the model. First, TranT-M with $M = 2$ is better than $M = 1$ on all datasets with all N and backbone network settings. This shows that the tracker can benefit from multiple templates, especially when the tracker itself does not perform well, such as TranT-M with ResNet101. TranT-M records the historical appearance information in the online tracking process; thus, it is robust to appearance change. Second, we observe that the performance of $M = 3$ is better than $M = 1$ but slightly worse than $M = 2$. The reason is that more updated templates will create more inaccuracies, resulting in performance degradation. Thus, we choose $M = 2$ as the default.

### TABLE IV
Comparison With Different N of TranST

| Method | N | Param (M) | Model Time (ms) | Speed (fps) | LaSOT [43] | TrackingNet [44] | GOT-10k [45] |
|--------|---|-----------|-----------------|-------------|-------------|------------------|---------------|
|        |   |           |                 |             | AUC P NM  | AUC P NM        | AUC P NM      |
| TranST | 2 | 16.7      | 12              | 71          | 64.2       | 73.5             | 68.2          |
| TranST | 4 | 23.0      | 17              | 54          | 64.9       | 73.8             | 69.0          |
| TranST | 6 | 29.3      | 22              | 42          | 64.4       | 73.2             | 68.5          |

The best results are shown in the red font.

### TABLE V
Comparison With Different Backbones of TranST

| Method | Backbone | Param (M) | Model Time (ms) | Speed (fps) | LaSOT [43] | TrackingNet [44] | GOT-10k [45] |
|--------|----------|-----------|-----------------|-------------|-------------|------------------|---------------|
|        |          |           |                 |             | AUC P NM  | AUC P NM        | AUC P NM      |
| TranST | ResNet18 | 17.1      | 15              | 61          | 63.9       | 72.9             | 67.7          |
| TranST | ResNet50 | 23.0      | 17              | 54          | 64.9       | 73.8             | 69.0          |
| TranST | ResNet101| 42.0      | 21              | 43          | 64.2       | 72.8             | 68.2          |

The best results are shown in the red font.

### TABLE VI
Comparison With Different N and Different M of TranST-M

| Method | N | M | Param (M) | Model Time (ms) | Speed (fps) | LaSOT [43] | TrackingNet [44] | GOT-10k [45] |
|--------|---|---|-----------|-----------------|-------------|-------------|------------------|---------------|
|        |   |   |           |                 |             | AUC P NM  | AUC P NM        | AUC P NM      |
| TranT-M| 2 | 1 | 16.8      | 12              | 71          | 64.6       | 73.7             | 68.6          |
| TranT-M| 2 | 2 | 16.8      | 13              | 56          | 64.6       | 73.7             | 68.6          |
| TranT-M| 2 | 3 | 16.8      | 14              | 52          | 64.6       | 73.7             | 68.6          |
| TranT-M| 4 | 1 | 23.1      | 17              | 54          | 65.4       | 73.9             | 69.6          |
| TranT-M| 4 | 2 | 23.1      | 18              | 42          | 65.4       | 73.9             | 69.6          |
| TranT-M| 4 | 3 | 23.1      | 21              | 38          | 65.4       | 73.9             | 69.6          |
| TranT-M| 6 | 1 | 29.4      | 22              | 42          | 64.2       | 72.6             | 67.8          |
| TranT-M| 6 | 2 | 29.4      | 24              | 35          | 64.2       | 72.6             | 67.8          |
| TranT-M| 6 | 3 | 29.4      | 27              | 31          | 64.2       | 72.6             | 67.8          |

The best results are shown in the red font.

### TABLE VII
Comparison With Different Backbones and Different M of TranST-M

| Method | backbone | M | Param (M) | Model Time (ms) | Speed (fps) | LaSOT [43] | TrackingNet [44] | GOT-10k [45] |
|--------|----------|---|-----------|-----------------|-------------|-------------|------------------|---------------|
|        |          |   |           |                 |             | AUC P NM  | AUC P NM        | AUC P NM      |
| TranT-M| ResNet18 | 1 | 17.2      | 15              | 61          | 64.1       | 73.6             | 67.5          |
| TranT-M| ResNet18 | 2 | 17.2      | 16              | 50          | 64.1       | 73.6             | 67.5          |
| TranT-M| ResNet18 | 3 | 17.2      | 19              | 45          | 64.1       | 73.6             | 67.5          |
| TranT-M| ResNet50 | 1 | 23.1      | 17              | 54          | 65.4       | 73.9             | 69.6          |
| TranT-M| ResNet50 | 2 | 23.1      | 18              | 42          | 65.4       | 73.9             | 69.6          |
| TranT-M| ResNet50 | 3 | 23.1      | 21              | 38          | 65.4       | 73.9             | 69.6          |
| TranT-M| ResNet101| 1 | 42.1      | 21              | 43          | 63.1       | 72.8             | 65.9          |
| TranT-M| ResNet101| 2 | 42.1      | 22              | 38          | 63.1       | 72.8             | 65.9          |
| TranT-M| ResNet101| 3 | 42.1      | 25              | 35          | 63.1       | 72.8             | 65.9          |

The best results are shown in the red font.
setting. Finally, when $M = 1$, TransT-M achieves basically the same performance as TransT without retraining, making the application of TransT-M flexible. The users can freely select the value of $M$ to balance the performance and the speed without retraining.

**Inconsistency of Training-Testing Number of Templates.** As mentioned above, we train TransT-M with two templates. A natural question is whether different numbers of training templates influence the performance of different numbers of testing templates. For example, the three-template setup is worse than the two-template setup during testing, which may be caused by the inconsistency of the training-testing number of templates. If the impact of this inconsistency is small, then we can use the one-template setup for training to reduce the training overhead. To explore this issue, we conduct experiments with different numbers of training and testing templates. The results are reported in Table VIII. Here, we use $M_{\text{test}}$ and $M_{\text{train}}$ to replace $M$ to avoid confusion. $M_{\text{test}}$ is the number of templates used in testing, and $M_{\text{train}}$ is the number of templates used in training. From the results, first, we see that the single-template (1-template) training setup works well when the testing number is 1, but it cannot benefit from the multi-template testing. This is because the model trained with the single-template setup never learned the updated templates during training, which makes it cannot adapt to the multi-template testing. Thus, the training-testing inconsistency has a large influence on single-template training. Second, for the multi-template training (2-template and 3-template for training), we find that the 2-template setup for training achieves the best performance. We also observe that the 2-template testing is the best for both the 2-template and 3-template training setups. These results show that the effect of training-testing inconsistency appears to be small for multi-template setups, and the number of templates itself has a larger influence. Single-template training cannot adapt to multi-template testing, while 3-template training and testing reduce performance because of more imprecision. Overall, the 2-template training setup and the 2-template testing setup are good candidates in our experiments. Under this setting, our TransT-M model can take a different number of templates as input without retraining, and the users can freely choose to use the single-template setup for faster speed or the 2-template setup for better performance.

**Different Template Combination Methods.** By default, we use the concatenating method to combine the multi-template features (i.e., we concatenate the features in the spatial dimension before the feature fusion network). Here, we also attempt to use averaging instead of concatenating to combine the multi-template features to make our method more efficient. For the averaging method, we directly average the features of multiple templates and send them to the feature fusion network. The results are shown in Table IX (in this table, all variants adopt the 2-template training setup). `Concat` indicates that we concatenate multi-template features, and `Avg` indicates that we average multi-template features. For the 1-template testing, the methods of concatenating and averaging are equivalent; thus, we use `"-"` to ignore this part. The first two lines are our default methods with different numbers of testing templates, using the concatenating method for both training and testing. In the third line, we use the averaging method in testing while keeping the concatenating method in training. We observe that the averaging method performs worse than the concatenating method. In the fourth and fifth lines, we also change the training method to averaging. We find that using the averaging method in training further reduces performance. Thus, we choose the concatenating method to pursue better performance. Compared with the concatenating method, the main disadvantage of the averaging method is that it inevitably integrates unrelated feature vectors of different templates, although it saves some computational overhead.

**Segmentation Head.** We conduct an ablation study of the segmentation head on VOT2021, as shown in Table X. VOT2021 uses mask annotation for evaluation. Thus, TransT only achieves 30.0% EAO by reporting the bounding box. After adding the segmentation head (denoted as SEG), the TransT-SEG can achieve 50.7% EAO. With the multi-template scheme and the segmentation head, TransT-M further obtains 55.0% EAO. We remove the additional attention in the segmentation branch (denoted as SEG-na), and then the performance of TransT-M drops to 53.7%, as shown in the third line in Table X.

**E. Evaluation on Other Datasets**

We evaluate our tracker on some commonly used small-scale datasets, including NFS [70], OTB2015 [71], and UAV123 [72].
TABLE IX
COMPARISON WITH DIFFERENT TEMPLATE COMBINATION METHODS

| Method  | Train | Test | \(M_{test}\) | Param | Model Time (ms) | Speed (fps) | LaSOT | TrackingNet | GOT-10k |
|---------|-------|------|-------------|-------|----------------|-------------|--------|-------------|---------|
| TransT-M | Concat | - | 1 | 23.1 | 17 | 54 | 65.4 | 73.9 | 69.6 | 81.7 | 68.4 | 79.8 | 72.7 | 82.9 | 69.0 |
| TransT-M | Concat | Concat | 2 | 23.1 | 18 | 42 | 65.4 | 73.9 | 69.6 | 82.5 | 87.3 | 80.0 | 74.7 | 85.5 | 71.3 |
| TransT-M | Concat | Avg | 2 | 23.1 | 17 | 54 | 65.4 | 73.9 | 69.6 | 82.0 | 86.8 | 79.6 | 73.1 | 83.7 | 69.5 |
| TransT-M | Avg | - | 1 | 23.1 | 17 | 54 | 63.1 | 73.5 | 67.5 | 79.5 | 85.3 | 77.5 | 69.0 | 79.8 | 62.9 |
| TransT-M | Avg | Avg | 2 | 23.1 | 17 | 54 | 63.7 | 73.5 | 67.5 | 79.9 | 85.7 | 77.6 | 70.5 | 82.1 | 63.9 |

The best results are shown in the red font.

TABLE X
ABLAITION STUDY ON VOT2021

| Method  | Seg | EAO | A | R |
|---------|-----|-----|---|---|
| TransT | - | 30.0 | 47.0 | 76.8 |
| TransT-SEG | SEG | 50.7 | 74.8 | 81.7 |
| TransT-M | SEG-na | 53.7 | 73.9 | 86.2 |
| TransT-M | SEG | 55.0 | 74.2 | 86.9 |

The best results are shown in the red font.

TABLE XI
COMPARISON WITH THE STATE-OF-THE-ART ON THE OTB, NFS, AND UAV123 DATASETS IN TERMS OF OVERALL AUC SCORE

| Method | NFS [70] | OTB [71] | UAV123 [72] |
|--------|----------|----------|-------------|
| TransT-M | 66.2 | 68.9 | 70.9 |
| TransT | 65.7 | 69.4 | 69.1 |
| TrDiMP [27] | 66.5 | 71.9 | 67.5 |
| STTTrack [52] | - | 71.1 | 64.7 |
| SiamGAT [54] | - | 71.0 | 64.6 |
| SiamBAN-ACM [53] | - | 72.0 | 64.8 |
| DStRPn [55] | - | 64.6 | - |
| PrDiMP [14] | 63.5 | 69.6 | 68.0 |
| DiMP [12] | 62.0 | 68.4 | 65.3 |
| SiamRPN++ [6] | 50.2 | 69.6 | 61.3 |
| ATOM [5] | 58.4 | 66.9 | 64.2 |
| HP-siam [73] | 52.1 | 60.1 | - |
| HP-basi [73] | 50.3 | 63.4 | - |
| ECO [26] | 46.6 | 69.1 | 53.2 |
| MDNet [62] | 42.2 | 67.8 | 52.8 |

The best three results are shown in red, blue, and green fonts.

V. CONCLUSION

In this work, we propose a novel, simple, and high-performance tracking framework based on the transformer-like feature fusion network. The proposed network conducts feature fusion solely using the attention mechanism, which includes an ego-context augment module based on self-attention and a cross-feature augment module based on cross-attention. The attention mechanism establishes long-distance feature associations, making the tracker adaptively focus on useful information and extract abundant semantic information. The proposed fusion network can replace the correlation to fuse the template and search region features, thereby facilitating object localization, bounding box regression, and instance segmentation. In addition, a segmentation branch, a multi-template scheme and an IoU prediction head are developed to further improve the tracker’s performance. Numerous experimental results on many benchmarks show that the proposed tracker performs significantly better than the state-of-the-art methods while running at a real-time speed.

Limitation. One limitation of our tracker is that, despite the high performance, it shows difficulty in predicting accurate bounding boxes when the target and distractors are approached. Some failure cases are shown in Fig. 10. Second, our method is developed on a GPU with high computing power and may not run at real-time speed on edge devices, thus limiting its practical application. Third, while we use the transformer as the feature fusion network, our backbone still employs the CNN-based network. Recently, some works [74], [75], [76], [77], [78] demonstrate that employing the transformer as the backbone network can further improve performance. Moreover, although we design a new architecture for tracking, we do not

We also collect some state-of-the-art and baseline trackers for comparison. The results are shown in Table XI.

NFS: We evaluate the proposed trackers on the 30 fps version of NFS [70] dataset. The NFS dataset contains challenging videos with fast-moving objects. Our TransT-M and TransT achieve the second- and third-best performance, respectively.

OTB2015: OTB2015 [71] contains 100 sequences and 11 challenge attributes. Table XI shows that our method does not achieve top performance on this dataset. We note that OTB2015 is a small-scale dataset and is easily overfitted, and we have not deliberately tuned the hyperparameters for this dataset.

UAV123: UAV123 [72] includes 123 low-altitude aerial videos and adopts success and precision metrics for evaluation. As shown in Table XI, the proposed method performs the best.
pretrain the model. Thus, the potential of the model may not be fully exploited.

Future Work. In future work, we are interested in improving our model’s efficiency and capacity through the following methods: 1) Employing transformers as the backbone network, 2) developing pretraining methods for tracking, 3) designing lightweight models for edge devices, and 4) incorporating explicit modules to handle distractors.

REFERENCES

[1] P. Li, D. Wang, L. Wang, and H. Lu, “Deep visual tracking: Review and experimental comparison,” Pattern Recognit., vol. 76, pp. 323–338, 2018.

[2] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaee, “Deep learning for visual tracking: A comprehensive survey,” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 5, pp. 3943–3968, May 2022.

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, and P. H. S. Torr, “Fully-convolutional Siamese networks for object tracking,” in Proc. Eur. Conf. Comput. Vis. Workshops, 2016, pp. 850–855.

[4] B. Li, X. Yuan, W. Bai, and X. Hu, “High-performance visual tracking with Siamese region proposal network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8971–8980.

[5] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “ATOM: Accurate tracking by overlap maximization,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4660–4669.

[6] B. Li, W. Wu, Q. Wang, Z. Zhang, J. Xing, and J. Yan, “SiamRPN++: Evolution of Siamese visual tracking with very deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4282–4291.

[7] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, “SiamFC++: Towards robust and accurate visual tracking with target estimation guidlines,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 12 549–12 556.

[8] Y. Wang, H. Peng, F. Fu, B. Li, and W. Hu, “Ocean: Object-aware anchor-free tracking,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 771–787.

[9] J. Choi, J. Kwon, and K. M. Lee, “Deep meta learning for real-time target-aware visual tracking,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 911–920.

[10] Y. Yu, Y. Xiong, W. Huang, and M. R. Scott, “Deformable Siamese attention networks for visual object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6728–6737.

[11] F. Du, P. Liu, W. Zhao, and X. Tang, “Correlation-guided attention for corner detection based visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6836–6845.

[12] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, “Learning discriminative model prediction for tracking,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 6182–6191.

[13] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, and F. S. Khan, “Learning the model update for Siamese trackers,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 4010–4019.

[14] M. Danelljan, L. V. Gool, and R. Timofte, “Probabilistic regression for visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 7183–7192.

[15] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 6000–6010.

[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 91–99.

[17] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “ECO: Efficient convolution operators for tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6638–6646.

[18] N. Wang, W. Zhou, J. Wang, and H. Li, “Transformer meets tracker: Exploiting temporal context for robust visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1571–1580.

[19] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, “Learning spatio-temporal transformer for visual tracking,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 10 448–10 457.

[20] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Annu. Conf. North Am. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol., 2019, pp. 4171–4186.

[21] C. Lüscher et al., “RWTH ASR systems for LibriSpeech: Hybrid versus attention,” in Proc. Int. Speech Commun. Assoc., 2019, pp. 231–235.

[22] G. Synnaeve et al., “End-to-end ASR: From supervised to semi-supervised learning with modern architectures,” 2019, arXiv:1911.08460.

[23] N. Parmar et al., “Image transformer,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 4055–4064.

[24] T. Y. Lin et al., “Microsoft COCO: Common objects in context,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 740–755.

[25] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, and J. Y. Choi, “Attentional correlation filter network for adaptive visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4807–4816.

[26] R. Tao, E. Gavves, and A. W. M. Smeulders, “Siamese instance search with Siamese region proposal network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 8343–8443.

[27] X. Chen, B. Yan, J. Zhu, D. Wang, and H. Lu, “Transformer meets tracking: Exploiting temporal context for robust visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 6836–6846.

[28] Z. Zha, W. Wu, W. Zhou, and J. Yan, “End-to-end flow correlation tracking with spatial-temporal attention,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 548–557.

[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[30] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2117–2125.

[31] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. D. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 658–666.

[32] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully convolutional neural networks for volumetric medical image segmentation,” in Proc. Int. Conf. 3D Vis., 2016, pp. 556–557.

[33] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2980–2988.

[34] H. Fan et al., “LaSOT: A high-quality benchmark for large-scale single object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 5374–5383.

[35] M. Muller, A. Bibi, S. Giancola, S. Alsabahi, and B. Ghanem, “TrackingNet: A large-scale dataset and benchmark for object tracking in the wild,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 300–317.

[36] L. Huang, X. Zhao, and K. Huang, “GOT-10k: A large high-diversity benchmark for generic object tracking in the wild,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 5, pp. 1562–1577, May 2021.

[37] C. Mayer et al., “Transforming model prediction for tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 8731–8740.

[38] Z. Song, J. Yu, Y.-P. Chen, and W. Yang, “Transformer tracking with cyclic shifting window attention,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 8791–8800.
[59] I. Jung, M. Kim, E. Park, and B. Han, “Online hybrid lightweight representations learning: Its application to visual tracking,” in Proc. Int. Joint Conf. Artif. Intell., 2022, pp. 1002–1008.

[60] M. Guo et al., “Learning target-aware representation for visual tracking via informative interactions,” in Proc. Int. Joint Conf. Artif. Intell., 2022, pp. 927–934.

[61] F. Tang and Q. Ling, “Ranking-based Siamese visual tracking with time-space memory networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 1374–1383.

[62] W. Han, X. Dong, F. S. Khan, L. Shao, and J. Shen, “Learning to fuse asymmetric feature maps in Siamese trackers,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 16570–16580.

[63] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, and C. Shen, “Graph attention tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 9543–9552.

[64] J. Shen, Y. Liu, X. Dong, X. Lu, F. S. Khan, and S. C. Hou, “Distilled Siamese networks for visual tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 12, pp. 8986–8990, Dec. 2022.

[65] P. Voigtlaender, J. Luiten, P. H. S. Torr, and B. Leibe, “Siam R-CNN: Visual tracking by re-detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6578–6588.

[66] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, “Know your surroundings: Exploiting scene information for object tracking,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 205–221.

[67] L. Zheng, M. Tang, Y. Chen, J. Wang, and H. Lu, “Learning feature embeddings for discriminant model-based tracking,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 759–775.

[68] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng, “Tracking by Instance Detection: A meta-learning approach,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6288–6297.

[69] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, “SiamCAR: Siamese fully convolutional classification and regression for visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6269–6277.

[70] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, “Siamese box adaptive network for visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 6668–6677.

[71] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 4293–4302.

[72] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, pp. 211–252, 2015.

[73] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proc. Int. Conf. Artif. Intell. Statist., 2010, pp. 249–256.

[74] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–10.

[75] L. Lin, H. Fan, Y. Xu, and H. Ling, “SwinTrack: A simple and strong baseline for transformer tracking,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 16743–16754.

[76] B. Ye, H. Chang, B. Ma, S. Shan, and X. Chen, “Joint feature learning and relation modeling for tracking: A one-stream framework,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 341–357.

[77] B. Chen et al., “Backbone is all you need: A simplified architecture for visual object tracking,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 375–392.

[78] Y. Cui, C. Jiang, L. Wang, and G. Wu, “MixFormer: End-to-end tracking with iterative mixed attention,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2022, pp. 13608–13618.
Xiang Ruan received the BE degree from Shanghai Jiao Tong University, Shanghai, China, in 1997, and the ME and PhD degrees from Osaka City University, Osaka, Japan, in 2001 and 2004, respectively. From 2007 to 2016, he was an expert engineer with OMRON Corporation, Kyoto, Japan. He is currently the Founder and CEO of Tiwaki Co., Ltd., Kusatsu, Japan. His current research interests include computer vision, machine learning, and image processing.

Dong Wang received the BE degree in electronic information engineering and the PhD degree in signal and information processing from the Dalian University of Technology (DUT), Dalian, China, in 2008 and 2013. He is currently a full professor with the School of Information and Communication Engineering, DUT. His current research interests include object detection and tracking.