Received 12 November 2023, accepted 25 December 2023, date of publication 1 January 2024, date of current version 23 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3348509

RESEARCH ARTICLE

AtptTrack: Asymmetric Transformer Tracker With Prior Templates

HAO ZHANG, YAN PIAO, NAN QI, AND YUE WANG
College of Electronic Information Engineering, Changchun University of Science and Technology, Changchun 130000, China

Corresponding author: Yan Piao (piaoyan@cust.edu.cn)

This work was supported by the Natural Science Foundation of Jilin Province under Grant 20220201062GX.

ABSTRACT Recently, Siamese-based trackers have emerged as the predominant focus in single object tracking research. However, the majority of these works concentrate on improving the backbone network of the tracker to enhance its performance, thereby overlooking the significant impact that the template and search region of the input to the tracker have on tracking accuracy. To address the aforementioned issues, we propose an Asymmetrical Transformer Tracker with Prior Templates (AtptTrack), consisting of a tracking branch and a template update branch. The function of the tracking branch is to receive input image pairs and tracking results to complete the tracking task. In the template update branch, an updating strategy is employed to compute the cosine similarity between the template and the tracking result. Based on this, four prior templates are generated, serving as essential supplementary features for the template. These prior templates are concatenated with the tracking results to create a hybrid template for subsequent tracking, enhancing the richness and accuracy of the template features. To further enrich the information content of the template and search region, we propose multi-scale patch embeddings to process input image pairs, which can enhance the completeness and continuity of the object features. Our tracker has been extensively tested on five benchmarks. The experiments demonstrate that our tracker achieves the state-of-the-art performance. Particularly on the OTB100 dataset, our tracker AtptTrack achieves an AUC score of 0.709, and it outperformed the second-place tracker in the deformation and occlusion challenges by 2.99% and 0.5%, respectively.

INDEX TERMS Visual tracking, transformer, Siamese network, template update.

I. INTRODUCTION

Visual Object Tracking (VOT), which involves accurately locating a specified target within successive video frames, stands as one of the significant research directions in the field of computer vision [1], [2], [3]. Recently, owing to the outstanding performance of Siamese-based trackers [4], [5], [6], [7], [8], they have gradually become the mainstream model for single object tracking research. The tracker is designed based on a parallel architecture of the Siamese network, which initially provides a template, and then specifies an area in the subsequent videos as the search region. Ultimately, the target is located by matching the features of the template in the search region.

SiamFC [4] represents the pioneering work in Siamese-based trackers, with several studies focusing on enhancing the tracker's performance by improving the backbone network of SiamFC. Trackers such as SiamRPN++ [5], SiamFC++ [6], and SiamGAT [7] have adopted ResNet50 [9] or GoogleNet [10] in place of AlexNet [11], which was used in SiamFC. It is worth mentioning that ViT [12] designed a target detector using Transformer [13] and demonstrated excellent performance in the field of computer vision. Therefore, some advanced trackers [14], [15], [16], [17], [18], adopting ViT as the backbone network are called Transformer-based tracker.

However, we believe that the accuracy of the trackers not only depends on the choice of the backbone network, but more importantly, on how to provide richer template features to the tracker. Some existing works have designed

The associate editor coordinating the review of this manuscript and approving it for publication was Shovan Barma.

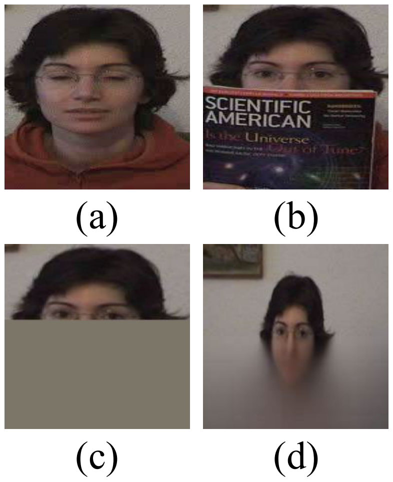

FIGURE 1. **MAE-based reconfiguration experiments. (a) and (b) represent the original images, (c) is the image simulating the target occlusion, and (d) is the reconstructed image.**

the template update network for the tracker [16], [17], [18], to enhance the information content of template features.

These works merely utilize information from the temporal dimension, employing different strategies to use the tracking result of a certain frame as the new template. Nevertheless, when the tracking results are not sufficiently accurate, a discrepancy can arise between the new template image and the initial template, leading to tracking failure. In other words, these works essentially employ a replacement strategy for altering the template, without truly enriching the feature information of the template.

In this paper, we design a separate template update branch for the tracker, which can provide templates with prior knowledge. The inspiration for the design of the template update branch is drawn from MAE [52], which stands as one of the pivotal contributions in self-supervised learning. MAE designs an asymmetric encoder-decoder architecture. The input and output of MAE are a masked image and a reconstructed image respectively. The goal is to achieve high similarity between the input image and the reconstructed image. As shown in FIGURE 1, we observed that MAE's capability to reconstruct input images can effectively fade out obstacles. We selectively mask a region in (a) to emulate the occlusion seen on the girl's face in (b), producing the result shown in (c). Subsequently, when (c) is input into the MAE, we obtain the reconstructed image (d). Notably, (d) not only reconstructs the obscured section of the girl's face but also eliminates the obstructions. Based on our above experiments and discussions, we believe that the reconstructed image serves as an additional source of a priori information. In tracking tasks, merging the template with the reconstructed image yields a template enhanced with prior knowledge, providing richer features than the original template alone.

On the other hand, the input for the Transformer-based tracker comprises two images of different sizes: the template and the search region. As the appearance of the tracking target can change, relying on a single patch size for patch embeddings may not ensure the integrity of the target's features. Consequently, we propose multi-scale patch embeddings to process the input image pairs, ensuring a more comprehensive representation of the target features. We designed patches of two distinct sizes to split the images. The smaller patches capture more intricate details, while the larger patches ensure the continuity of feature information. By combining these, we can obtain a more complete feature representation.

In summary, the main contributions of this paper are:

- We design an individual template update branch for our tracker, which leverages asymmetric encoder-decoder architecture to generate the priori templates. This approach effectively fades out obstacles and represents the features of the occluded target. Subsequently, the input template and the priori templates are concatenated to obtain the hybrid template. In comparison with existing trackers that provide a single template, our approach furnishes enriched template information, thereby bolstering tracking precision and robustness.
- We propose a multi-scale patch embeddings for processing input image pairs, which provides the tracker with a more detailed input representation. Both the template and the search region are split into patches of varied sizes. By leveraging complementary features, our approach ensures continuous and comprehensive representation of target features.
- Our tracker achieves state-of-the-art performance on several challenging benchmarks. Notably, it achieves an AUC score of 0.703 on the LaSOT [19] dataset.

II. RELATED WORKS

In single object tracking tasks, mainstream trackers can be broadly categorized into three classes: DCF-based trackers, Siamese-based trackers and Transformer-based trackers. Notably, Transformer-based trackers represent the latest research direction, as a result, generally outperform other trackers in performance.

A. DCF-BASED TRACKER

DCF-based trackers operate by online training of a high-performance correlation filter to accomplish target tracking. SRDCF [20] introduces spatially regularized discriminative correlation filters, allowing it to accept more negative samples for training, thereby enhancing the discriminative capability of the tracker. BACF [21] proposes a background-aware correlation filter based on hand-crafted features. It can simulate the variations in the target foreground and background over time, enabling it to effectively utilize negative samples for filter updates. CFNet [22] combines correlation filters with Siamese networks to design an asymmetric correlation filter tracker. Differing from other DCF-based trackers, it employs training images to update the correlation filter, enriching the positive and negative samples used for filter updates, thereby enhancing the tracker's robustness.

Although DCF-based trackers have shown excellent performance, they come with an inherent limitation that cannot be overlooked. Their heavy reliance on manually designed correlation filters and the number of positive and negative samples used for filter updates compromises the generalization ability of DCF-based trackers.

B. SIAMESE-BASED TRACKER

In contrast to DCF-based trackers, the performance of Siamese-based trackers primarily relies on the backbone network. With the rapid advancements in convolutional neural networks, the design of Siamese-based trackers has become more concise and stable, resulting in better performance. Consequently, Siamese-based trackers have emerged as the prevailing research direction in current single-object tracking.

In recent years, researchers have employed methods such as reinforcement learning [26], [27], unsupervised learning [28], [29], and graph convolution [30] to enhance trackers, achieving commendable performance. Other works have focused on modifying the backbone network of the tracker to enhance performance. SiamFC is the pioneering work on Siamese-based trackers. The tracker primarily receives two inputs: the template, specified from the object of interest in the initial frame of the video, and the search region extracted from subsequent frames of the video sequence. Then, the input image pairs are fed into the modified AlexNet for feature extraction to obtain two feature maps. Finally, the feature maps are used to perform a cross-correlation computation, producing a score map that indicates the location of the target. Inspired by the region proposal network (RPN) [31], SiamRPN [32] introduces a region proposal sub-network into the tracker. This sub-network encompasses both classification and regression branches, enabling foreground-background differentiation and refined target proposal suggestions. Building upon this, DaSiamRPN [33] introduces a disturbance-aware module to address interference from semantic backgrounds to the foreground, and SiamFC++ proposes guidelines for target state estimation, which meticulously classifies different states of the target, achieving more precise localization. SiamBAN [34] and SiamCAR [35] refine the bounding box regression method for SiamRPN, leading to more accurate and authentic bounding box outputs from the regression branch. SiamRPN++ addresses the limitation of Siamese-based trackers being unable to leverage deeper backbone networks through a simple spatial aware sampling strategy. It replaces the AlexNet in SiamRPN with a deeper ResNet50, resulting in a significant enhancement in the tracker's feature extraction capabilities. Owing to the more tangible performance enhancements achieved by improving the backbone networks of trackers, many studies have delved deeply into this direction. For instance, ATOM [36] and DIMP [37] employ ResNet18 [9] as their backbone network, while SiamGAT [7] adopts GoogleNet for feature extraction. These trackers have also achieved commendable evaluation results.

C. TRANSFORMER-BASED TRACKERS

As Transformers have exhibited impressive performance in the realm of natural language processing [40], [41], ViT has expanded its applications to include object detection. Consequently, many excellent ViT-based detectors [42], [48], [49], were proposed. In the Swin Transformer [42], a shifted window strategy is introduced to address the issue of unfolding non-overlapping patches. This innovation not only enhances computational efficiency but also offers higher-resolution feature information for downstream tasks. MAE combines a self-supervised training method with ViT to complete the pixel-level image reconstruction task. It begins by applying a mask to a certain portion of the input image and subsequently reconstructs the masked image through an asymmetric encoder-decoder architecture. Specifically, the encoder facilitates feature interactions among the masked patches, with "masked" referring to the random dropping of a specific proportion of patches. On the other hand, the decoder's role is to realign these masked patches to their original sequence and then perform pixel-level reconstruction, aiming to obtain a reconstructed image that closely mirrors the input.

Given the notable efficacy of these frameworks, several studies [14], [15], [43], [44], [45], have incorporated them into trackers to enhance feature extraction capabilities. TranT [14] combines ResNet50 and Transformer to design a hybrid backbone network, introducing the ego-context augment module and the cross-feature augment module to enhance the semantic feature extraction capability of the tracker. TrSiam [43] combines ResNet50 with the encoder and decoder of Transformer, establishing distinct and parallel pipelines. In these pipelines, the encoder branch handles multiple frames of the tracking template to reinforce the temporal context of the template features, while the branch with the decoder carries out feature matching for target localization.

SwinTrack [15] employs the Swin-transformer as the backbone network of the tracker and utilizes the encoder-decoder architecture of Transformer to achieve feature fusion. Distinctly different from TransT and TrSiam, SwinTrack does not adopt a hybrid backbone framework but instead designs a pure Transformer-based tracker. OSTrack [44] and SimTrack [45] propose a more concise and intuitive one-stream framework. Following embedding processing, the template and search region are concatenated and passed through a pre-trained model based on MAE/ViT for feature extraction. The resulting output is then fed to the prediction head. On the other hand, some studies [16], [17], [18], [46], [47], have introduced template update networks. For instance, TrTr [18] introduces a convolution-based online update module to address the challenges of target deformation. STARK [16] proposes a spatio-temporal feature fusion approach to dynamically selecting tracking results as new templates.

Based on the aforementioned and discussions, we believe that refining the backbone network of the tracker can lead to significant performance enhancement. However, the impact of the template and search area on the performance the tracker should not be overlooked. When the backbone network of the tracker is sufficiently robust, the more comprehensive the information contained in the input image, the more precise the tracking becomes. Therefore, we propose the AtptTrack, which comprises a localization branch dedicated to the tracking task and a template update branch designed to provide enriched template features.

III. PROPOSED METHOD

In this section, we introduce the proposed tracker AtptTrack, which encompasses both a tracking branch and a template update branch. Specifically, the tracking branch is primarily responsible for the classification and regression of objects, while the template update branch provides richer prior knowledge to the template.

A. OVERVIEW OF THE TRACKER

As shown in FIGURE 2, the tracker utilizes the Transformer to design two distinct branches with an asymmetric parallel architecture. The input of the tracker aligns with that of typical Transformer-based trackers, encompassing both a template and a search region. In the tracking branch, a multi-scale patch embeddings method is employed to project the input image pairs into embedded tokens, which are then passed to the backbone network based on the MAE pre-trained Transformer encoder. Subsequently, the template tokens and search region tokens output from the backbone are concatenated and subjected to dimension reduction. These refined tokens are then fed into the feature interaction module to execute feature matching and information interaction. Ultimately, the outputs from this module are directed to the prediction head to perform regression and classification.

On the other hand, in the template update branch, the tracker updates the template at specified frames according to the template update strategy. Initially, the tracking results from the tracking branch are fed into the template update branch and undergo a linear projection, generating new target template tokens. Then, a certain proportion of these tokens is masked and passed through a Transformer encoder for feature encoding. Notably, this encoder shares weights with the backbone network in the tracking branch. The output from the encoder is then fed to a Transformer decoder to retrieve prior template tokens. Finally, tokens from both components are concatenated to produce hybrid template tokens enriched with prior knowledge. These hybrid template tokens are reintroduced into the tracking branch as the updated template for object tracking.

B. TRACKING BRANCH

Patch Embeddings: Firstly, we split and unfold the input image pairs $z \in \mathbb{R}^{3\times H_z\times W_z}$ and $x \in \mathbb{R}^{3\times H_x\times W_x}$ into 2D patches $z \in \mathbb{R}^{N_z\times(P^2\times 3)}$ and $x \in \mathbb{R}^{N_x\times(P^2\times 3)}$. This unfolding operation is accomplished through a convolutional layer, where (P, P) represents the size of the convolution kernel, P is the stride of the convolutional layer, while $N_z = H_z \times W_z/P^2$ and $N_x = H_x \times W_x/P^2$ denote the number of patches for the template image and search region image respectively. After that, a trainable linear projection layer is employed to map the 2D patches into 1D patch embeddings with a dimension of C. Finally, to ensure the ordered nature of the patch embeddings, learnable 1D position embeddings are used to supplement both template patches and search region patches. The resulting template tokens $Z \in \mathbb{R}^{N_z\times C}$ and search region tokens $X \in \mathbb{R}^{N_x\times C}$ serve as inputs to the backbone network.

Pipeline of Tracking Branch: The tracking branch primarily consists of three components: the backbone network, the feature interaction module and the prediction head. Previous studies [44], [45] have demonstrated that pre-trained models offer superior initialization parameters for the tracker's backbone network, facilitating easier convergence during training and significantly reducing development time. Consequently, we adopt the MAE-based pre-trained Transformer encoder as the backbone network of our tracker, whose function is to generate feature representations for both the template and search region. It's worth noting that designing pipelines for tracking tasks requires a balance between real-time and performance considerations. In MAE experiments, the best performing network is the ViT-Large, encompassing 24 Transformer blocks with parameters of about 300M. Despite its exceptional performance, it undeniably demands substantial computational resources. Therefore, we employ the ViT-base, which is more suitable for tracking tasks. The ViT-Base consists of 12 Transformer blocks. Each block contains multi-head self-attention (MHSA), multi-layer perceptron (MLP) and layer normalization (LN), interconnected with residual connections. After processing through l-th Transformer block, the template tokens and search region tokens are denoted as Z^l and X^l respectively:

$$\begin{aligned} Z^{l'}, X^{l'} &= MHSA\left(LN\left(Z^l, X^l\right)\right) + \left(Z^l, X^l\right), \\ Z^{l+1}, X^{l+1} &= MLP\left(LN\left(Z^{l'}, X^{l'}\right)\right) + \left(Z^{l'}, X^{l'}\right), \end{aligned} \tag{1}$$

where, Z^{l+1} and X^{l+1} are denoted as the outputs of l-th Transformer block.

The multi-head attention (MHA) is critical for implementing the Transformer block, taking in three inputs: Query (Q), Key (K), and Value (V). Notably, when Q, K, and V originate from the same source, it is referred to as MHSA. The mathematical representation of attention is as follows:

$$Attention\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

where, $1/\sqrt{d_k}$ is the scaling factor.

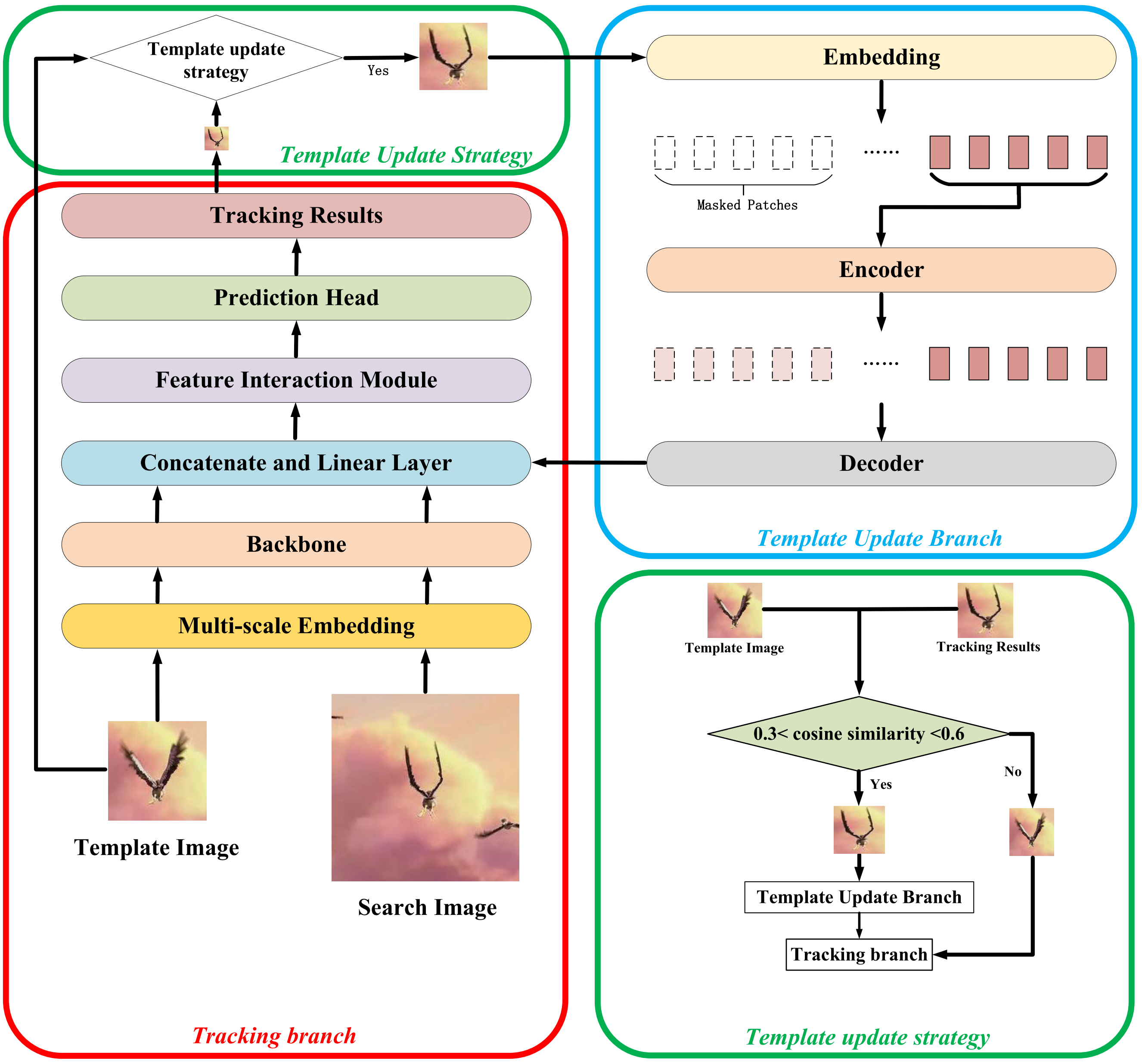

FIGURE 2. Overview of the tracker architecture, including a tracking branch, a template update branch and a template update strategy.

The MHA maps the input vector into N matrices of dimension D, where N denotes the number of heads in the MHA. The specific mathematical expression is as follows:

$$MHA\,(Q,K,V) = Concat\,(H_1,H_2,\ldots,H_N)\,W^O, \\ where\ H_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \quad (3)$$

where, $W_i^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}}\times d_v}$ and $W^O \in \mathbb{R}^{N\times d_v\times d_{\text{mod}\,el}}$ represent parameter matrices. In our experiments, $L = 12$, $N = 16$ and $d_{\text{model}} = 768$.

C. FEATURE INTERACTION MODULE

As shown in FIGURE 3, the feature interaction module consists of three Transformer blocks and a linear projection layer. We concatenate the outputs of the three blocks into a prediction vector, which serves as the input to the prediction head. Distinct from prior works [14], [44], we employ residual connections to fuse the outputs of the Transformer blocks, thereby enhancing the information content of the feature interaction module.

The feature interaction module partially receives outputs $Z_{\text{out}} \in \mathbb{R}^{N_z\times C}$ and $X_{\text{out}} \in \mathbb{R}^{N_x\times C}$ from the template and search region processed by the backbone network, while another part receives outputs Z_{tb} from the tracking results processed through the template update branch. First, we concatenate Z_{out} and X_{out} to obtain the feature interaction tokens, and then, a 1×1 convolutional layer φ_1 is utilized to map its dimensions to C_1. The mathematical

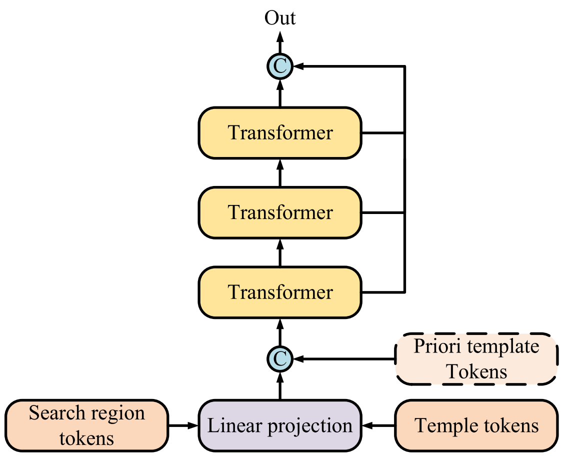

FIGURE 3. Overview of the feature interaction module. It is worth noting that the prior template tokens are concatenated only during the template update process.

expression is:

$$T_{out} = Concat\left(Z_{out}, X_{out}\right)$$
$$T_{out}^{c_1} = \phi_1\left(T_{out}\right) \tag{4}$$

where $T_{out} \in \mathbb{R}^{(N_z+N_x)\times C}$ and $T_{out}^{c_1} \in \mathbb{R}^{(N_z+N_x)\times C_1}$. In our experiments, $C = 768$ and $C_1 = 512$.

Secondly, T_{out} and Z_{tb} are concatenated in the 0-th dimension to produce $T_{concat}^{c_1}$:

$$T_{concat}^{c_1} = Concat\left(T_{out}^{c_1}, Z_{tb}\right) \tag{5}$$

where $Z_{tb} = Concat\left(Z_t, Z_b, Z_l, Z_r\right)$ and $T_{concat}^{c_1} \in \mathbb{R}^{(5\times N_z+N_x)\times C_1}$.

Finally, $T_{concat}^{c_1}$ are fed into the feature interaction module. The outputs from each Transformer block within the module are concatenated along 2-th dimension, producing feature interaction tokens used for prediction head.

$$T_{\text{concat}} = Concat\left(\varphi_1\left(T_{concat}^{c_1}\right), \varphi_2\left(\varphi_1\left(T_{concat}^{c_1}\right)\right), \varphi_3\left(\varphi_2\left(\varphi_1\left(T_{concat}^{c_1}\right)\right)\right)\right) \tag{6}$$

where $T_{\text{concat}} \in \mathbb{R}^{(5\times N_z+N_x)\times(C_1\times 3)}$, and φ_1 (), φ_2 () and φ_3 () represent each Transformer block respectively. Specifically, in our experiments, the number of heads for the MHSA in these Transformer blocks is 16, with an intermediate dimension of 512.

D. PREDICTION HEAD AND LOSS FUNCTION

The prediction head is a feedforward neural network (FNN) consisting of L blocks, where each block comprises a convolutional layer, layer normalization and an activation layer (GELU). The input to the prediction head is the prediction feature map $T_{pfm} \in \mathbb{R}^{\left(\frac{N_x}{P}\times\frac{N_x}{P}\right)\times 256}$, which is a reshaped vector of T_{concat}. The classification sub-branch of the prediction head is responsible for predicting the foreground or background, while the regression sub-branch is employed to estimate the probabilities of the top-left and bottom-right corners of the bounding box.

During training, the input consists of two randomly selected frames within a specified range. The pair of input images is processed through the tracker to perform feature extraction and interaction, after which it is fed into the prediction head. We employ a weighted focal loss for foreground and background classification. For bounding box regression, we use the L_1 loss combined with the GIoU loss [50]. The loss function is as follows:

$$L = L_{cls} + \lambda_{iou}L_{iou} + \lambda_{L_1}L_1 \tag{7}$$

where $\lambda_{iou} = 2$ and $\lambda_{L_1} = 5$ are hyper-parameters for the loss function. The setting of the hyper-parameters follows [16] and [44]. This configuration has been proven effective in our experiments, hence we retained this setup.

E. TEMPLATE UPDATE BRANCH

We design a template update branch to generate the priori templates for the tracker, which can provide richer complementary features to the template by simulating target occlusion. The architecture of the template update branch is illustrated in FIGURE 1. It consists of an encoder for feature representation, a decoder for pixel-level reconstruction and a template update strategy.

The input to the template update branch is either the given initial template Z_0 or the tracking result of a specific frame Z_n. First, we process the input image into tokens using patch embeddings. After that, we selectively mask the top, bottom, left and right corners of the input image, simulating four scenarios of target occlusion. Specifically, taking the simulation of the upper half of the target being occluded as an example, we assume the input image is $I \in \mathbb{R}^{H_i\times W_i\times 3}$ and the patch size is P_i. Thus, the input image is mapped into embedding tokens $T_i \in \mathbb{R}^{N_i\times C_i}$, where $N_i = \frac{H_i}{P_i} \times \frac{W_i}{P_i}$. The tokens in the interval $[0, \frac{N_i}{2}]$ are masked, resulting in the prior template tokens representing the scenario where the upper half is occluded. The process for other occlusion simulations follows a similar pattern. Finally, the four prior template tokens are fed into the template update branch for encoding and decoding to reconstruct the occluded tokens. Notably, the encoder shares weights with the backbone network of the localization branch. The decoder employs eight Transformer blocks, each with 16 heads and an intermediate dimension of 512.

As shown in FIGURE 4 , we visualize the templates and the prior templates. We selected four templates and generated a prior template for each of them. On the one hand, when the updated template contains obstructions, the prior template can effectively supplement pure target features.

As can be seen from (11) and (12), obstacles such as the car and trees obstructing the bus are attenuated in the reconstruction process, which is crucial for presenting the complete features of the template. On the other hand, when there are no obstacles in the template, yet obstacles are encountered in subsequent tracking.

As shown in FIGURE 5, we input the template, search region and prior template into the pretrained backbone network of our tracker and visualize the feature maps output by

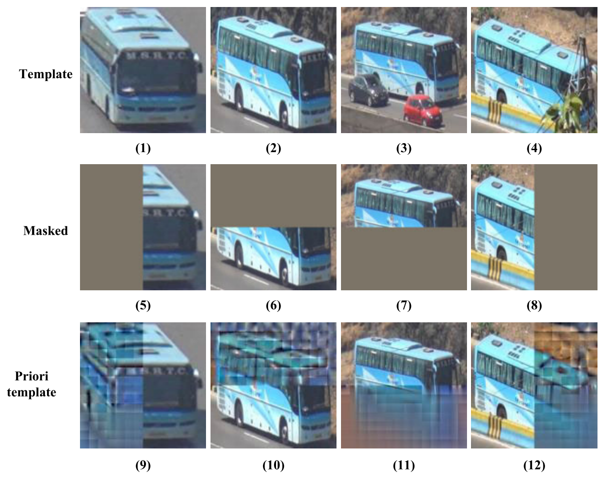

FIGURE 4. Visualization of templates and a priori templates.

the last layer. By comparing (G) with (H) and (J) with (K), it is evident that the template and the prior template display inconsistent target feature information. In other words, the features extracted in the presence of target obstruction closely resemble those displayed by the prior template. Specifically, as shown in (H) and (I), when the car is unobstructed, the backbone network focuses more on the car's headlights. When the car is obscured, the attention is predominantly directed towards the car's body, closely resembling the target feature presented in the prior template. Likewise, when the tiger's tail is obscured, the network focuses more on the tiger's head, which is closer to the target feature presented in (K). In summary, we believe that the prior template plays a pivotal role when the target is obscured. It serves as crucial supplementary information for the template, contributing to the enhancement of continuity and accuracy in tracking.

Template Update Strategy: In tracking tasks, the object does not exhibit significant changes across consecutive video frames. Therefore, we refrain from updating the template in every frame. Instead, we introduce a template update strategy to ensure both the accuracy and efficiency of the tracker, as shown in FIGURE 1.

During the tracking process, the template $z_{n-1} \in \mathbb{R}^{H_z \times W_z \times 3}$ and the search region $x_{n-1} \in \mathbb{R}^{H_x \times W_x \times 3}$ are processed using the patch embeddings method $E()$, resulting in $Z_{n-1} \in \mathbb{R}^{N_z \times C}$ and $X_{n-1} \in \mathbb{R}^{N_x \times C}$. These are then fed into the tracking branch $\tau_{\text{tracking}}()$ to obtain the tracking result $o_n \in \mathbb{R}^{H_o \times W_o \times 3}$, which is scaled to match the same size as the Z_{n-1} and mapped by $E()$ to tracking result tokens O_n. Finally, the cosine similarity between the template and the tracking result tokens is calculated. The mathematical expression is as follows:

$$
\begin{aligned}
o_n &= \tau_{\text{tracking}}\left(z_{n-1}, x_{n-1}\right) \\
Z_{n-1} &= \mathrm{E}\left(z_{n-1}\right), O_n = \mathrm{E}\left(o_n\right) \\
C &= \cos\left(Z_{n-1}, O_n\right) \\
\textit{where } \cos\left(Z_{n-1}, O_n\right) &= \frac{Z_{n-1} \cdot O_n}{\| Z_{n-1} \| \| O_n \|}
\end{aligned} \tag{8}
$$

where $C \in [0, 1]$ represents the similarity between the template and the tracking result.

When the cosine similarity is small, we believe that the target may be heavily occluded; therefore, we do not update the template. Conversely, when the cosine similarity is large, we believe that the current template is accurate and, again, do not update the template. However, when $C \in [0.3, 0.5]$, indicating that the target may undergo deformation or is largely occluded, we update the template in this interval to ensure the accuracy of the template.

We use the tracking result o_n as the new template, and it is fed into both the template update branch and the backbone network of the tracking branch to generate hybrid template tokens. Firstly, in the tracking branch, the tracking result is processed by the backbone network $\phi_b()$ to produce new template tokens $O_n \in \mathbb{R}^{N_c \times C}$. A 1×1 convolutional layer $\varphi_1()$ is then employed to map the dimensionality of O_n to $O' \in \mathbb{R}^{N_z \times C_1}$. Meanwhile, the template update branch $\tau_{\text{updata}}()$ processes the tracking result to get prior template tokens Z_u, Z_d, Z_r, Z_l from four angles. These four prior template tokens are then concatenated with the new template tokens to create hybrid template tokens Z_{h}. Finally, the search region x_n is processed through the backbone network of the tracker $\phi_b()$ to produce the search tokens X_n. These search tokens are concatenated with the hybrid template tokens and fed to the feature interaction module to accomplish the object tracking task. The mathematical expression is as follows:

$$
\begin{aligned}
Z_u, Z_d, Z_l, Z_r &= \tau_{updata}\left(o_n\right) \\
O'_n &= \varphi_{linear}\left(\phi_b\left(o_n\right)\right) \\
Z_h &= Concat\left(\left[O'_n, Z_u, Z_d, Z_l, Z_r\right]\right) \\
T_{out} &= Concat\left(\left[Z_h, X_n\right]\right)
\end{aligned} \tag{9}
$$

where $Z_h \in \mathbb{R}^{(5 \times N_z) \times C_1}$ and $T_{out} \in \mathbb{R}^{(5 \times N_z + N_x) \times C_1}$.

IV. MULTI-SCALE PATCH EMBEDDINGS

While ViT employs a single patch size for patch embeddings, this approach is suitable for object detection. The majority of images in the ImageNet-1K dataset [51] are static with distinct target, allowing a single size to sufficiently extract target features. Many Transformer-based trackers [42], [44], [45] have adopted this methodology. However, the size and shape of targets in tracking tasks are unpredictable. To address the aforementioned issue, we propose a multi-scale patch embeddings approach designed to better align with the diverse characteristics of targets in tracking tasks. This method aims to optimize the input preprocessing of the tracking branch.

We employ the serialization method from Section III-B to preprocess the input template $x_n \in \mathbb{R}^{W_x \times H_x \times 3}$ and search region $z \in \mathbb{R}^{W_z \times H_z \times 3}$ to obtain $X \in \mathbb{R}^{\frac{W_x}{P} \times \frac{H_x}{P} \times C}$ and $Z \in \mathbb{R}^{\frac{W_z}{P} \times \frac{H_z}{P} \times C}$. In our approach, P has two sizes, $P_1 = 16$ and $P_2 = 32$. Therefore, the outputs of the multi-scale patch embeddings are $X' \in \mathbb{R}^{\left(\frac{W_x}{P_1} \times \frac{H_x}{P_1} + \frac{W_x}{P_2} \times \frac{H_x}{P_2}\right) \times C}$ and

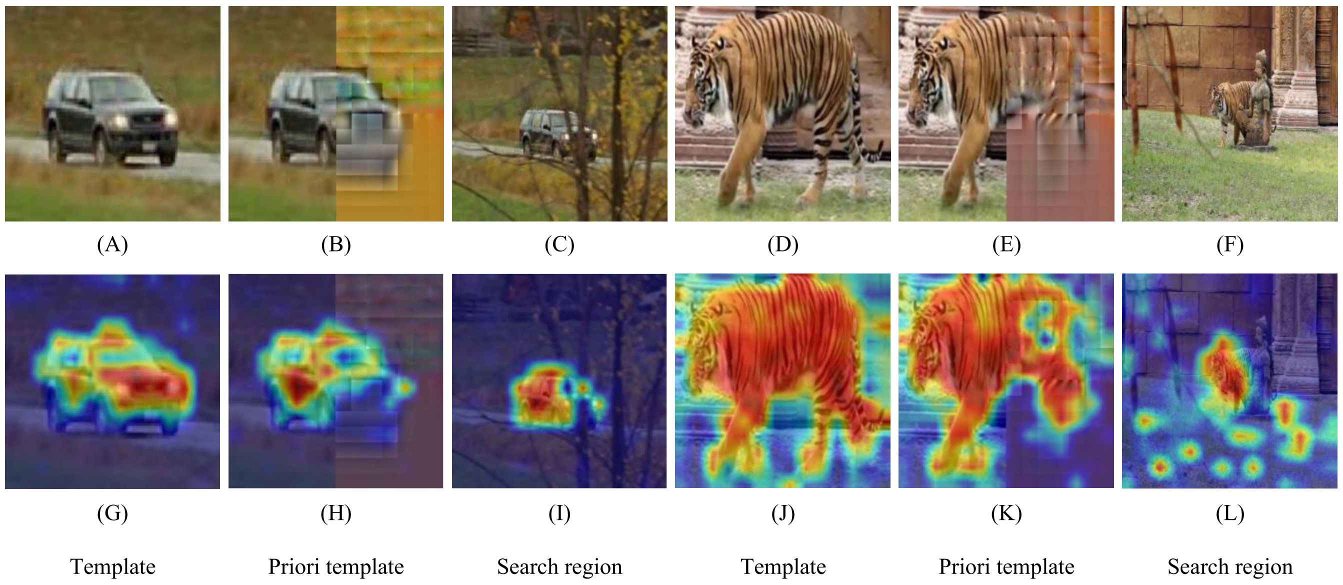

FIGURE 5. **Visualization of feature maps. The feature maps are extracted from the last layer of the backbone network.**

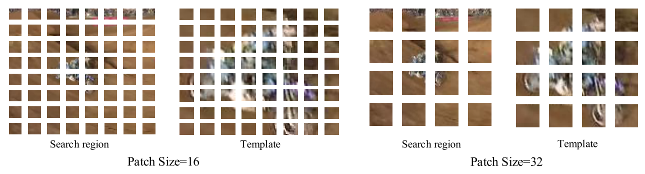

FIGURE 6. **Multi-scale patch embeddings method, which divides input image of the tracking branch into patches using various sizes.**

$Z' \in \mathbb{R}^{\left(\frac{W_z}{P_1}\times\frac{H_z}{P_1}+\frac{W_z}{P_2}\times\frac{H_z}{P_2}\right)\times C}$, where $C = 768$. As shown in FIGURE 6, our approach excels in extracting intricate features of the target while simultaneously preserving feature integrity. Within the smaller patch size, distinct features of the rider and the motorcycle are clearly delineated. Meanwhile, in the larger patch size, there is enhanced coherence in representing the target's features. The wheels of a motorcycle are extracted more comprehensively.

V. EXPERIMENTS

A. IMPLEMENTATION DETAILS

Our approach is implemented using Python and PyTorch. The models are trained on 4 NVIDIA 3090 GPUs and inference speed is evaluated on single NVIDIA 3090 GPU. First, the vanilla ViT-Base [12] pre-trained with MAE as the backbone network of the tracker. Subsequently, the decoder and multi-scale patch embeddings are trained using the methods in MAE. It is important to note that we freeze the decoder's parameters in subsequent tracking training. For tracking training, we utilize diverse datasets, including LaSOT [19], GOT-10K [53], TrackingNet [54] and COCO [55]. The template and search region are set to 128×128 and 256×256 respectively. The model is trained using the AdamW [56]. Specifically, the learning rate for the backbone network is set at $5e{-}5$, while other components are set at $5e{-}4$. Our training epoch is 600 with a batch size of 128. And the learning rate undergoes a reduction by a factor of ten at the $200-th$ and $400-th$ epochs.

B. COMPARISON WITH TRACKERS

Comparative experiments between our tracker and a number of representative trackers in five benchmark challenges show that our tracker achieves state-of-the-art results.

Results on OTB100: The OTB100 dataset contains 100 video sequences, encompassing 11 different categories of tracking challenges, including occlusion, deformation, motion blur, fast motion, scale variations, background clutters, low resolution, in-plane rotation, illumination variations, out-of-view, and out-of-plane rotation. Additionally, it evaluates tracker performance based on precision and success rate. Specifically, the center distance between the predicted tracking results and the ground truth is determined and compared against a predefined threshold to ascertain accuracy. Also, the intersection over union (IoU) between the tracking predictions and the ground truth is computed; an IoU exceeding a particular threshold signifies successful tracking.

The area under the curve (AUC) of the success plot serves as an indicator of the overall tracking efficacy.

As shown in FIGURE 7, our tracker AtptTrack achieves state-of-the-art performance in the OTB100 dataset. AtptTrack obtains an AUC score of 0.709, marking a 1.14% improvement over the second-ranked method. Furthermore, AtptTrack demonstrated excellent results in the occlusion and deformation challenges of the OTB100 dataset. In the deformation challenge, AtptTrack's AUC score surpasses the second-place result by 2.99%. Similarly, in the occlusion challenge, there is a performance enhancement of 0.5%.

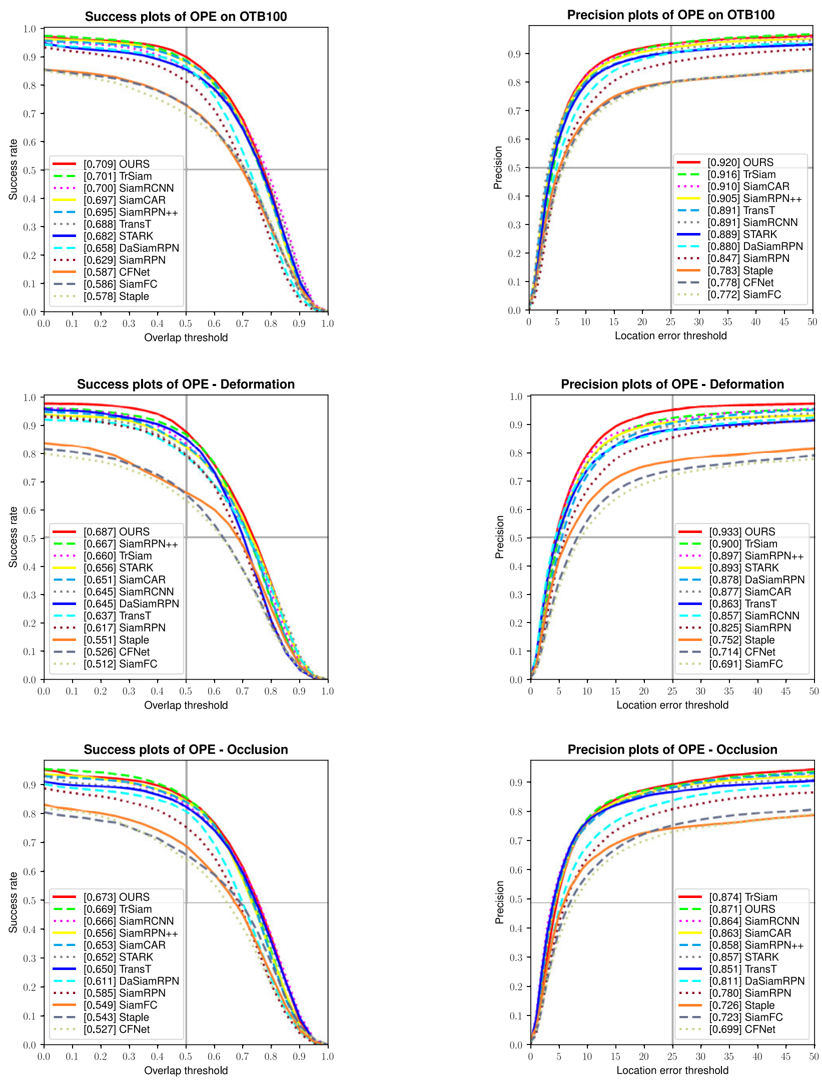

FIGURE 7. We conduct comparative experiments on the OTB100 dataset and visualize the AUC and accuracy curves.

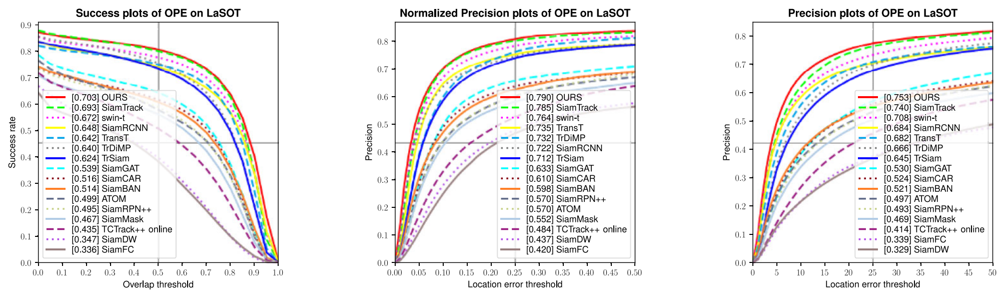

FIGURE 8. Comparison results of our tracker with several excellent trackers on the LaSOT dataset.

Results on LaTOT: The LaSOT dataset is a large-scale and long-term tracking dataset consisting of 1,400 video sequences across 70 categories. The video sequences in the LaSOT dataset have a maximum length of 11,397 frames, a minimum of 1,000 frames, and an average length exceeding 2,500 frames. Evaluation criteria of the LaSOT dataset include AUC, precision, and normalized precision. As shown in FIGURE 8, our tracker AtptTrack achieves an AUC score of 0.703 on the LaSOT dataset, surpassing Transformer-based SimTrack and Siamese-based SiamRCNN [57] by 1.4% and

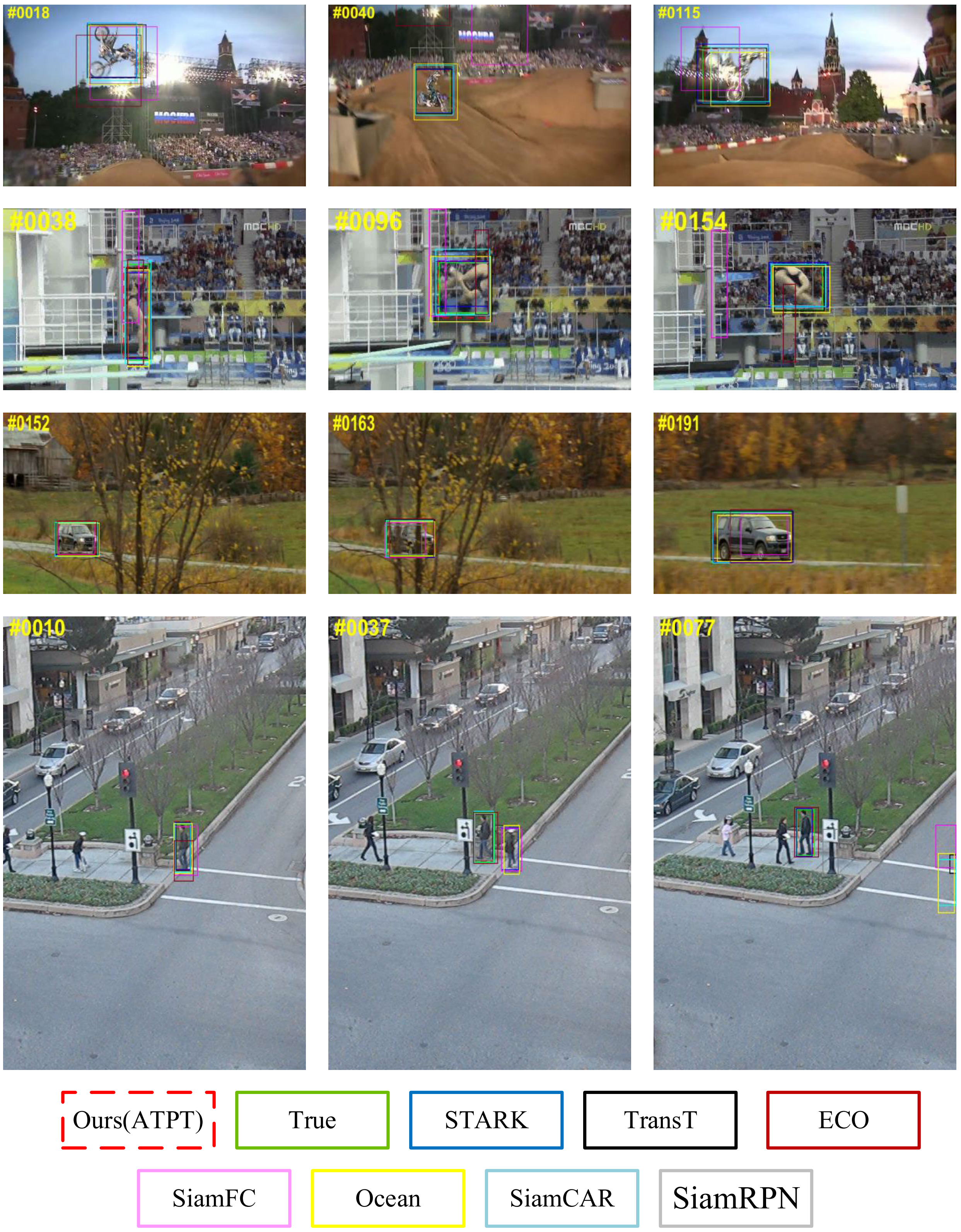

FIGURE 9. Qualitative experiments. Our tracker AtptTrcker is compared with trackers STARK, TransT, ECO, SiamFC, Ocean, SiamCAR, and SiamRPN on four video sequences from the OTB100 dataset, and the tracking results are visualized.

8.4% respectively. It is noteworthy that our tracker with 125M parameters run at 36*fps* on the LaSOT dataset, while the top-performing Siamese-based SimRCNN runs at 5*fps*. This demonstrates the strong competitiveness of our tracker.

Results on GOT-10k: GOT-10K is a large-scale dataset comprising 10K videos for training and 180 videos for testing. We exclusively utilize the GOT-10K dataset for both training and evaluation of our tracker. As shown in Table 1, our tracker AtptTrack achieves the best performance across all three evaluation metrics on the GOT-10K dataset.

TABLE 1. Experiment results on GOT-10K.

Tracker	GOT-10K		
	AO ↑	$SR_{0.5}$ ↑	$SR_{0.75}$ ↑
SiamFC	34.8	35.3	9.8
SiamRPN	46.3	54.9	25.3
ATOM	55.6	63.4	40.2
SiamRPN++	51.7	61.6	32.5
SiamCAR	56.9	67.0	41.5
Ocean	61.1	72.1	47.3
SiamR-CNN	64.9	72.8	59.7
SiamGAT	62.7	74.3	48.8
TransT	67.1	76.8	60.9
START-S50	67.2	76.1	61.2
TrSiam	66.0	76.6	57.1
CSWinTT	69.4	78.9	65.4
OSTrack	71.0	80.4	68.2
SimTrack	68.6	78.9	62.4
AtptTracker	71.3	80.9	68.6

Results on VOT2018: The VOT2018 dataset, composed of 60 video sequences totaling 21,356 frames, is a widely used benchmark. This dataset introduces a reset mechanism that allows for the re-initialization of trackers after five frames of tracking failure, thereby enhancing the efficiency of its utilization. To evaluate tracker performance, the VOT2018 dataset employs the accuracy (A), robustness (R), and expected average overlap (EAO) metrics. Accuracy measures the IoU between the tracking results and the ground truth, while robustness quantifies the percentage of tracking failure frames. EAO is derived based on the overlap between tracking results and ground truth, serving as an estimation of the tracker's average precision. As shown in Table 2, our tracker achieves the top results in both the A and EAO evaluation metrics on the VOT2018 dataset, improving by at least 1.5% and 1.8% compared to other trackers.

Results on UAV123: The UAV123 dataset consists of 123 video sequences captured by unmanned aerial vehicles at low altitudes. A distinguishing feature of the UAV123 dataset is its aerial viewpoint of tracking targets. Such perspectives present targets that are not only smaller in size but also prone to frequent changes, thereby escalating the challenge of tracking. To evaluate tracker performance, the AUC is utilized by UAV123. As shown in Table 3, our tracker AtptTrack achieves an AUC score of 68.5, the second highest among all trackers.

TABLE 2. Experiment results on VOT2018.

Tracker	VOT2018		
	A ↑	R ↓	EAO↑
SRDCF	0.490	0.970	0.188
SiamFC	0.503	0.585	0.188
ECO	0.484	0.276	0.280
DaSiamRPN	0.586	0.276	0.383
SiamRPN++	0.604	0.234	0.415
ATOM	0.590	0.204	0.401
SiamFC++	0.556	0.183	0.400
Ocean	0.592	0.117	0.489
TrTr-offline	0.612	0.234	0.424
DiMP-50	0.597	0.153	0.440
TrDiMP	0.600	0.141	0.462
SiamAttn	0.630	0.160	0.470
AtptTracker	0.640	0.121	0.498

TABLE 3. Experiment results on UAV123.

Tracker	UAV123
	AUC ↑
SiamFC	49.2
ECO	52.5
SiamRPN++	64.2
ATOM	61.7
SiamCAR	61.4
SiamFC++	61.8
SiamGAT	64.6
DiMP	65.4
TransT	68.1
TrSiam	67.4
TrDiMP	67.5
OSTrack	68.3
SimTrack	69.8
AtptTracker	68.5

TABLE 4. Ablation experiments on the effect of different components on our tracker.

	Feature interaction module	Template update branch	Multi-scale patch embeddings	AUC (LaSOT)
①	✓	×	×	0.671
②	✓	✓	×	0.695
③	✓	✓	✓	0.703

C. ABLATION EXPERIMENTS

We conduct ablation experiments and analysis on the LaSOT dataset. The performance gains resulting from different components being added to the tracker are shown in Table 4.

Ablation experiments use ① as a benchmark, representing the tracking branch of our tracker. The architecture of ① is a regular Transformer-based tracker, similar to SimTrack and OSTrack. Therefore, it serves as an effective baseline to highlight the improvements brought about by various components. First, by incorporating the template updating branch into ①, we derive ②. Comparing the AUC scores between ① and ②, we observe a performance improvement of 3.6%, attributed to the richer prior features provided by the template updating branch. This suggests that as the backbone network reaches its optimal performance, introducing more input information can enhance performance. Subsequently, in ②, we incorporate multi-scale parch embeddings for input preprocessing, resulting in the AtptTrack, referred to as ③. Comparing ② and ③, it's evident that patches of varying sizes can achieve a performance gain of 1.2%, consistent with the insights from the comparison between ① and ②.

TABLE 5. Ablation experiments on template update strategy. *C* represents the cosine similarity computed using the template and the tracking result.

	Template update strategy			Speed (FPS)	AUC (LaSOT)
	$C \in [0.3, 0.5]$	$C \in [0.5, 0.8]$	$C \in [0.3, 0.8]$		
④	✓			36	0.703
⑤		✓		21	0.675
⑥			✓	13	0.705

Table 5. illustrates the impact of employing different thresholds for cosine similarity in the template updating strategy. Upon comparing ④ and ⑤, it is apparent that as the threshold for C increases, both the speed and performance of the tracker decline. We believe that this decline stems from the redundant updating of the template during the tracking process. In other words, when the template and the tracking result are similar, yet the template still undergoes an update, it affects not only the tracker's speed but also fails to boost its performance. Similarly, as the range of C expands, while there might be a slight growth in performance, the tracker can no longer operate in real-time. We regard such gains as counterproductive. In summary, by comparing ④, ⑤ and ⑥, it's evident that at $C \in [0.3, 0.5]$, the template undergoes updates under more justifiable conditions, achieving a balance between the performance and speed of AtptTrack.

D. QUALITATIVE COMPARISIONS

As shown in FIGURE 9, we present a qualitative comparison of our tracker with other trackers on the OTB100 dataset. Each row in the figure demonstrates the prediction results of different trackers for a given video sequence, including AtptTrack, STARK, TransT, ECO, SiamFC, Ocean, SiamCAR, and SiamRPN. Specifically, we purposely provide several sequences where the target has different challenges. For instance, the sequences displayed in the second and fourth rows showcase target deformation and occlusion, respectively. The outcomes demonstrate that our tracker consistently provides precise predictions across diverse scenarios, such as standard conditions, occlusions, and deformations. At the same time, the predicted bounding boxes closely mirror the ground truth.

VI. CONCLUSION

In this paper, we improve both the input preprocessing and the backbone network of the tracker. We propose AtptTrack, which encompasses a tracking branch, a template update branch and a prediction head. Notably, the multi-scale patch embeddings and the template update branch enrich the feature information of the template and search region, thereby bolstering the tracker's performance. In addition, the prior templates provided by the template update branch simulate target occlusion scenarios, serving as prior knowledge to effectively mitigate the adverse effects of occlusions on tracking accuracy. However, the tracker's performance is less than optimal in scenarios with small target tracking or fast motion. Our future research will focus on refining the tracker to address these challenges.

REFERENCES

[1] Y. Zhang, Q. Lin, H. Tang, and Y. Li, "Research on Siamese object tracking algorithm based on knowledge distillation in marine environment," *IEEE Access*, vol. 11, pp. 50781–50793, 2023.

[2] D. Huang, M. Yang, J. Duan, S. Yu, and Z. Liu, "Siamese network tracking based on feature enhancement," *IEEE Access*, vol. 11, pp. 37705–37713, 2023.

[3] C. Guo, D. Yang, C. Li, and P. Song, "Dual Siamese network for RGBT tracking via fusing predicted position maps," *Vis. Comput.*, vol. 38, no. 7, pp. 2555–2567, Jul. 2022.

[4] M. Cen and C. Jung, "Fully convolutional Siamese fusion networks for object tracking," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*. Cham, Switzerland: Springer, Oct. 2018, pp. 3718–3722.

[5] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of Siamese visual tracking with very deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4277–4286.

[6] Y. Xu, Z. Wang, Z. Li, Y. Yuan, and G. Yu, "SiamFC++: Towards robust and accurate visual tracking with target estimation guidelines," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 7, pp. 12549–12556.

[7] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, and C. Shen, "Graph attention tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9538–9547.

[8] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1328–1338.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[14] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8122–8131.

[15] L. Lin, H. Fan, Z. Zhang, Y. Xu, and H. Ling, "SwinTrack: A simple and strong baseline for transformer tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 16743–16754.

[16] 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. (ICCV)*, Oct. 2021, pp. 10428–10437.

[17] X. Hu, H. Liu, Y. Hui, X. Wu, and J. Zhao, "Transformer feature enhancement network with template update for object tracking," *Sensors*, vol. 22, no. 14, p. 5219, Jul. 2022.

[18] M. Zhao, K. Okada, and M. Inaba, "TrTr: Visual tracking with transformer," 2021, *arXiv:2105.03817*.

[19] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5369–5378.

[20] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4310–4318.

[21] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1144–1152.

[22] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5000–5008.

[23] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6931–6939.

[24] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 621–629.

[25] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1430–1438.

[26] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 105–114.

[27] N. Wang, W. Zhou, G. Qi, and H. Li, "Post: Policy-based switch tracking," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12184–12191.

[28] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Unsupervised deep tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1308–1317.

[29] N. Wang, W. Zhou, Y. Song, C. Ma, W. Liu, and H. Li, "Unsupervised deep representation learning for real-time tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 400–418, Feb. 2021.

[30] J. Gao, T. Zhang, and C. Xu, "Graph convolutional tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4644–4654.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 91–99.

[32] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with Siamese region proposal network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8971–8980.

[33] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware Siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 101–117.

[34] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese box adaptive network for visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6667–6676.

[35] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "SiamCAR: Siamese fully convolutional classification and regression for visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6268–6276.

[36] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate tracking by overlap maximization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4655–4664.

[37] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6181–6190.

[38] S. Yang, H. Chen, F. Xu, Y. Li, and J. Yuan, "High-performance UAVs visual tracking based on Siamese network," *Vis. Comput.*, vol. 38, no. 6, pp. 2107–2123, Jun. 2022.

[39] C. Li, S. Lin, J. Qiao, and S. An, "Partial tracking method based on Siamese network," *Vis. Comput.*, vol. 37, no. 3, pp. 587–601, Mar. 2021.

[40] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Sys.*, vol. 33, 2020, pp. 1877–1901.

[41] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[42] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002.

[43] N. Wang, W. Zhou, J. Wang, and H. Li, "Transformer meets tracker: Exploiting temporal context for robust visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1571–1580.

[44] 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.* Cham, Switzerland: Springer, 2022, pp. 341–357.

[45] B. Chen, P. Li, L. Bai, L. Qiao, Q. Shen, B. Li, W. Gan, W. Wu, and W. Ouyang, "Backbone is all your need: A simplified architecture for visual object tracking," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 375–392.

[46] L. Zhang, A. Gonzalez-Garcia, J. V. D. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for Siamese trackers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4009–4018.

[47] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic Siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1781–1789.

[48] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 213–229.

[49] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," 2020, *arXiv:2010.04159*.

[50] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 658–666.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[52] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 15979–15988.

[53] 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.

[54] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV.*, 2014, pp. 740–755.

[55] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "TrackingNet: A large-scale dataset and benchmark for object tracking in the wild," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 300–317.

[56] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[57] P. Voigtlaender, J. Luiten, P. H. S. Torr, and B. Leibe, "Siam R-CNN: Visual tracking by re-detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6577–6587.

[58] Z. Zhang and H. Peng, "Deeper and wider Siamese networks for real-time visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4586–4595.

[59] Z. Fu, Z. Fu, Q. Liu, W. Cai, and Y. Wang, "SparseTT: Visual tracking with sparse transformers," 2022, *arXiv:2205.03776*.

[60] F. Xie, C. Wang, G. Wang, W. Yang, and W. Zeng, "Learning tracking representations via dual-branch fully transformer networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 2688–2697.
[61] Z. Song, J. Yu, Y. P. Chen, and W. Yang, "Transformer tracking with cyclic shifting window attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8781–8790.
[62] Z. Zhang, H. Peng, J. Fu, B. Li, and W. Hu, "Ocean: Object-aware anchor-free tracking," in *Proc. 16th Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 771–787.
[63] Y. Yu, Y. Xiong, W. Huang, and M. R. Scott, "Deformable Siamese attention networks for visual object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6727–6736.

HAO ZHANG was born in 1993. He is currently pursuing the Ph.D. degree with the Changchun University of Science and Technology. His research interests include object tracking and object detection.

YAN PIAO was born in 1965. She received the M.S. and Ph.D. degrees from the Changchun Institute of Optics, Machinery and Precision. She is currently a Professor with the Changchun University of Science and Technology. Her main research interests include deep learning and integral image.

NAN QI was born in 1995. She is currently pursuing the Ph.D. degree with the Changchun University of Science and Technology. Her research interest includes epileptic seizure prediction.

YUE WANG received the B.S. degree in electronic information science and technology from Jilin Agricultural Science and Technology University, in 2013, and the M.S. degree in information and communication engineering from the Changchun University of Science and Technology, in 2016. Her current main research interests include image processing and computer vision.