Electronic Research Archive

ERA, 33(6): 3638–3672.
DOI: 10.3934/era.2025162
Received: 24 March 2025
Revised: 31 May 2025
Accepted: 06 June 2025
Published: 12 June 2025

https://www.aimspress.com/journal/ERA

Research article

A multi-scale cyclic-shift window Transformer object tracker based on fast Fourier transform

Huanyu Wu[1], **Yingpin Chen**[1,2,*], **Changhui Wu**[1], **Ronghuan Zhang**[1] **and Kaiwei Chen**[1]

[1] School of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China
[2] Key Laboratory of Light Field Manipulation and System Integration Applications in Fujian Province, Minnan Normal University, Zhangzhou 363000, China

* **Correspondence:** Email: 110500617@163.com.

Abstract: In recent years, Transformer-based object trackers have demonstrated exceptional performance in object tracking. However, traditional methods often employ single-scale pixel-level attention mechanisms to compute the correlation between templates and search regions, disrupting object's integrity and positional information. To address these issues, we introduce a cyclic-shift mechanism to expand the diversity of sample positions and replace the traditional single-scale pixel-level attention mechanism with a multi-scale window-level attention mechanism. This approach not only preserves the object's integrity but also enriches the diversity of samples. Nevertheless, the introduced cyclic-shift operation heavily burdens storage and computation. To this end, we treat the attention computation of shifted and static windows in the spatial domain as convolution. By leveraging the convolution theorem, we transform the attention computation of cyclic shift samples from the spatial domain to element-wise multiplication in the frequency domain. This approach enhances computational efficiency and reduces data storage requirements. We conducted extensive experiments on the proposed module. The results demonstrate that the proposed module outperforms multiple existing tracking algorithms regarding performance. Moreover, ablation studies show that the method effectively reduces the storage and computational burden without compromising performance.

Keywords: object tracking; window attention; cyclic-shift; fast Fourier transform; Transformer tracker

1. Introduction

As an important research area in computer vision [1,2], visual object tracking (VOT) [3] has wide applications in drone tracking [4–6], intelligent driving [7,8], object recognition [9,10], military applications [11,12], and other fields. Its primary research involves modeling an object's appearance and other features in the initial frame of a video sequence and then accurately locating it in subsequent frames to complete the tracking task [13]. However, in practical applications, trackers often face various challenges [14], such as deformation, partial occlusion, fast movement, and background interference [15], which lead to cumulative errors and affect tracking performance [16].

Currently, mainstream object trackers include correlation filter-based trackers and deep learning-based trackers [17,18]. By transforming complex spatial-domain correlation operations into frequency-domain element-wise multiplications, correlation filter (CF) trackers [19] achieve high computational efficiency in object tracking. This method has made CF methods widely used in visual object tracking [20]. For example, KCF [21] significantly improved tracking speed and accuracy by utilizing the fast Fourier transform (FFT) [22]. However, in practical applications, the circulant matrices used in CF methods introduce boundary effects, leading to the unsatisfactory performance of CF methods. BACF [23] mitigates boundary effects by expanding the search area and cropping negative samples from the real world to address this issue. Despite the widespread attention to CF methods, these approaches have limitations, such as filter degradation and insufficient robustness [24]. In recent years, researchers have made many improvements by introducing deep convolutional neural networks to enhance tracking accuracy, such as TEDS [25], DeepABCF [26], and DeepBACF [27]. Nevertheless, most CF methods only employ the initial frame of the object as the sole template in the tracking process, leading to the tracker's limits in its ability to model the object's appearance.

Recently, deep learning-based object trackers have gradually become the dominant framework in the object tracking field [28]. For example, SiamFC [29] applies the Siamese network [30] structure to VOT. These methods extract features from the object and search region via an offline-trained neural network and measure their similarity by a correlation operator. SiamRPN [31] designs a region proposal network (RPN) to achieve precise object localization, enhancing tracking accuracy. Subsequently, SiamRPN++ [32] employs a spatially aware sampling strategy to alleviate translation invariance issues caused by padding operations. Despite their success, Siamese networks are limited by the local linear property of correlation operations, which lack the capacity for modeling long-range information dependencies [33]. Thus, the Siamese networks may lose contextual semantic information, which is significant for robust tracking [34].

To address these limitations [35], scholars have recently introduced Transformer [36,37] architectures, which possess strong capabilities for modeling long-range dependencies, leading to the development of various Transformer-based trackers [38–40]. For example, TransT [41] replaces the correlation operations in Siamese networks with self-attention and cross-attention mechanisms, demonstrating the potential of attention mechanisms in VOT [42]. However, TransT relies solely on the first frame's template, which limits its ability to capture real-time appearance changes of the object. STARK [43] introduces a dynamic template to exploit spatio-temporal information and turns the tracking task into a bounding box prediction task to overcome this issue. However, these methods rely on global self-attention for feature extraction, which is highly complex. To address this, the Swin Transformer [44] employs local window self-attention and a shifted window strategy. This approach reduces computational complexity to a linear level while maintaining efficiency. It has been applied in various VOT tasks, such as SwinTrack [45]. Inspired by the Swin Transformer, CSWinTT [46] employs a novel cross-window attention calculation between the template and search region. It replaces conventional pixel-level attention with multi scale window attention via cyclic-shift (CS) windows. This approach preserves object integrity and enriches positional information, effectively distinguishing the object from the background. Furthermore, CSWinTT adopts a multi-head, multi-scale attention mechanism to measure similarity between partitioned windows at specific scales, enhancing tracking accuracy. However, the CS strategy for each window introduces a substantial computational burden due to the large number of matrix multiplications, leading to unsatisfactory tracking speed. Therefore, optimizing the computational approach to reduce the calculation and storage resources required by the tracker is of great significance.

Motivated by the correlation filter theory, which shows the correlation operation between a matrix and a CS matrix is efficiently computed using FFT [47], a worthwhile question is whether the FFT can avoid the high storage and computational complexity caused by cyclic shifted samples in the spatial domain. Inspired by this, we propose a multi scale cyclic shift window Transformer object tracker based on fast Fourier transform (MCWTT). Unlike the CSWinTT, which directly performs CS matrix operations in the spatial domain, we introduce the FFT and design a frequency domain feature fusion block. It converts the calculation strategy from the spatial domain to the frequency domain regarding the matrix multiplication of CS windows as a correlation operation. Then, according to the convolution theorem, we transform the correlation operation into point-wise multiplication in the frequency domain, alleviating the problems of increased storage and computational costs caused by traditional CS operations. For the object location task, we borrow the idea of the STARK tracker, decoupling classification and regression. It presents a corner estimation network that predicts the object's top-left and bottom-right corner coordinates, improving the model's handling of scale changes from rapid movements. Finally, to boost the model's spatio-temporal information mining, we add a prediction head to assess the best candidate samples' reliability and obtain dynamic templates with temporal information.

For this paper, our contributions are summarized as follows.

(1) We propose a window-level attention mechanism to replace the traditional pixel-level attention mechanism, thereby avoiding the potential disruption of object's integrity and positional information caused by pixel-level attention mechanisms. Moreover, combined with the CS operation, this mechanism promotes information exchange across windows, enriches training sample diversity, and improves the model's adaptability in complex scenarios.

(2) We design a novel attention calculation strategy that first treats the attention calculation in the spatial domain as a matrix correlation operation. We then leverage the advantages of the fast Fourier transform (FFT) to convert these operations into frequency-domain element-wise multiplications. This approach significantly improves computational efficiency and reduces storage costs, providing a more efficient solution for real-time object-tracking tasks.

(3) Extensive experimental analysis of the proposed MCWTT module was conducted on multiple datasets, including the LaSOT [48], OTB100 [49], and UAV123 [50] datasets, with detailed comparisons against various existing mainstream trackers. The experimental results demonstrate that the MCWTT tracker outperforms most existing methods across multiple evaluation metrics, especially showing significant advantages in handling scale variations and fast-moving objects. Furthermore, ablation studies further indicate that the proposed multi-scale shifted window attention mechanism plays an important role in improving trackers' performance and also demonstrates the efficiency of the frequency domain attention mechanism.

The rest of this paper is organized as follows. Section 2 introduces the preliminary knowledge. Section 3 explains the design and implementation of the proposed module. Section 4 presents experimental results comparing the proposed module with other methods on multiple datasets, as well as ablation studies. Finally, Section 5 summarizes the main findings and outlines future work.

2. Preliminary knowledge

2.1. Correlation operations and fast computation of image processing

Extracting the object's features and providing accurate prediction information in the subsequent tracking process is a key objective in object tracking tasks [51]. The circulation matrix structure expands the object samples, allowing the tracker to utilize spatio-temporal information further to exploit the deep features fully. However, the large amount of sample data generated by circulant matrix operations is redundant. If these data are directly operated on in the spatial domain, it will result in a significant computational burden and high computational complexity. Therefore, leveraging the connection between convolution and circulant matrices and transforming the operations to element-wise frequency-domain multiplications via the convolution theorem, avoiding large matrix multiplications and inversion operations.

Two-dimensional image correlation operations are special convolutions. The operation is carried out through element-wise multiplication in the frequency domain and the inverse Fourier transform, as shown in Eq (2.1) below:

$$\boldsymbol{I}_1 \bigstar \boldsymbol{I}_2 = \mathbf{Conv}_2\left(\boldsymbol{I}_1, \bar{\boldsymbol{I}}_2\right) = \mathbf{real}\left\{\mathbf{ifft}_2\left(\hat{\boldsymbol{I}}_1 \odot \hat{\boldsymbol{I}}_2^*\right)\right\}, \tag{2.1}$$

where the symbol $\bigstar$ represents the two-dimensional correlation operation, $\mathbf{Conv}_2$ denotes the two-dimensional convolution operation. $\boldsymbol{I}_1, \boldsymbol{I}_2 \in \mathbb{R}^{\sqrt{\boldsymbol{N}} \times \sqrt{\boldsymbol{N}}}$, $\sqrt{\boldsymbol{N}}$ is an integer, $\bar{\boldsymbol{I}}_2$ represents two-dimensional reversed signal of $\boldsymbol{I}_2$. The calculation rule for the reversed signal of a two-dimensional signal is as follows: First, perform row-wise reversing on the matrix to obtain an intermediate matrix, and then perform column-wise reversing on this intermediate matrix. $\hat{\boldsymbol{I}}_1$ and $\hat{\boldsymbol{I}}_2$ represent the Fourier transform of $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, and $\hat{\boldsymbol{I}}_2^*$ represents the conjugate of $\hat{\boldsymbol{I}}_2$. The symbol $\odot$ indicates element-wise multiplication, $\mathbf{ifft}_2$ represents the two-dimensional inverse Fourier transform, and $\mathbf{real}$ represents the operation of taking the real part.

When Eq (2.1) is computed in the spatial domain, the memory space occupied is $\boldsymbol{N}^2 + \boldsymbol{N}$, and the computational complexity is $\mathcal{O}\left(\boldsymbol{N}^2\right)$. If the convolution theorem transforms spatial-domain correlation to frequency-domain element-wise multiplication, the module's memory requirements drop to $4\boldsymbol{N}$, and the computational complexity becomes $\mathcal{O}\left(8\boldsymbol{N}\log_2 \boldsymbol{N} + 4\boldsymbol{N}\right)$.

2.2. Attention mechanism

The attention mechanism is an essential component of the Transformer model architecture, helping the model extract and fuse features from images and adjust the tracking region to enable the tracker to more effectively distinguish between foreground and background interference and more accurately track the object.

Assume the input sequence $\boldsymbol{X}$ has a size of $\boldsymbol{X} \in \mathbb{R}^{N \times C}$, where N denotes the number of tokens, and C denotes the length of tokens. $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}$, $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V}$. $\boldsymbol{Q} \in \mathbb{R}^{N \times D}$, $\boldsymbol{K} \in \mathbb{R}^{N \times D}$, $\boldsymbol{V} \in \mathbb{R}^{N \times D}$. $\boldsymbol{W}^{Q} \in \mathbb{R}^{C \times D}$, $\boldsymbol{W}^{K} \in \mathbb{R}^{C \times D}$, and $\boldsymbol{W}^{V} \in \mathbb{R}^{C \times D}$ are linear transformation matrices. Given the query feature vector $\boldsymbol{Q}_i = \mathrm{Split}(\boldsymbol{Q}) \in \mathbb{R}^{N \times d_i}$, the key feature vector $\boldsymbol{K}_i = \mathrm{Split}(\boldsymbol{K}) \in \mathbb{R}^{N \times d_i}$, and the value feature vector $\boldsymbol{V}_i = \mathrm{Split}(\boldsymbol{V}) \in \mathbb{R}^{N \times d_i}$ for each head, in the attention mechanism, the attention is determined by the self-attention score matrix, which decides the degree of attention each part of the input sequence should receive. As shown in Eq (2.2), the attention matrix is written as follows:

$$\boldsymbol{A} = \frac{\boldsymbol{Q}_i \boldsymbol{K}_i^{\mathrm{T}}}{\sqrt{d_i}}. \tag{2.2}$$

where d_i denotes the dimension of each token in the head.

The Softmax function converts the normalized attention score matrix into probabilities, which are then applied to the value matrix to produce the final attention output. The result of the self-attention computation can be defined by the following equation:

$$\mathrm{Attention}(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i) = \mathbf{Softmax}(\boldsymbol{A})\boldsymbol{V}_i, \tag{2.3}$$

where i denotes the number of input sequence heads.

Using multi-head attention enhances the performance of the attention mechanism. The multi-head attention calculation is defined as follows:

$$\begin{gathered}\mathrm{Multihead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathbf{Concat}(\boldsymbol{H}_1, \boldsymbol{H}_2, \cdots, \boldsymbol{H}_I)\boldsymbol{W}^{O}, \\ \text{s.t. } \boldsymbol{H}_i = \mathrm{Attention}(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i)\end{gathered} \tag{2.4}$$

where I denotes the number of attention heads, $\boldsymbol{W}^{O} \in \mathbb{R}^{D \times D}$ is a linear transformation matrix, and $i \in [1, I]$.

3. Proposed module

In this section, we introduce the proposed multi-scale cyclic-shift window Transformer object tracker (MCWTT). The main framework of the module consists of four parts: the backbone network, the feature fusion block (FFB), the feature enhancement block (FEB), and the prediction analysis module. The architecture of the module is illustrated in Figure 1.

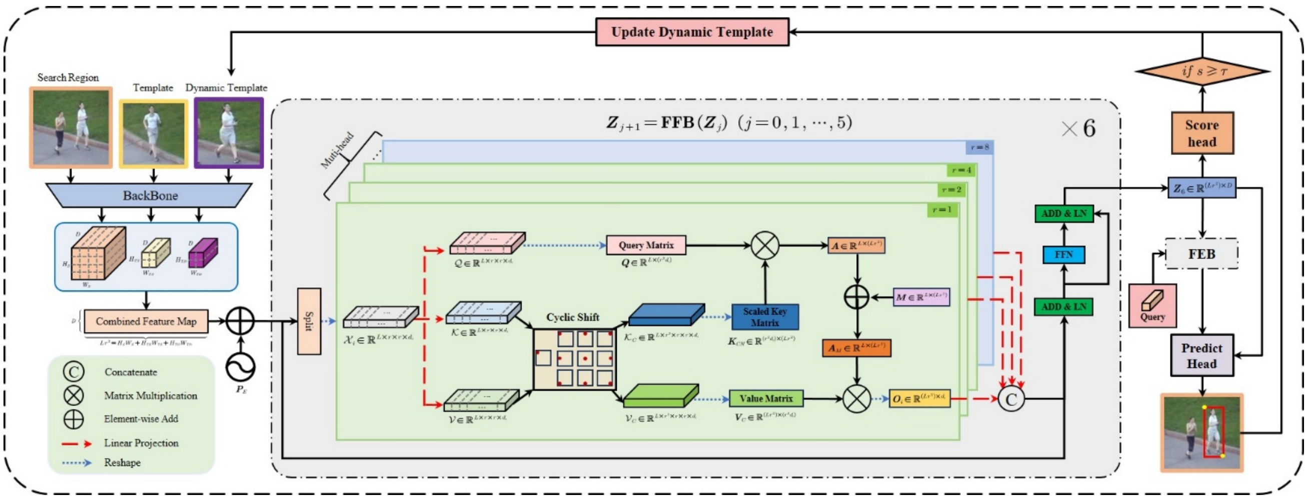


Figure 1. The overall architecture of the MCWTT module.

3.1. Backbone network

The backbone network of the tracker is based on ResNet-50 [52], which extracts features from the search region image $\mathcal{S}$, the initial static template image $\mathcal{T}_s$, and the dynamic template image in the initial frame $\mathcal{T}_d$. These inputs are fed into the backbone network to obtain the corresponding feature maps, namely the search region feature map $\mathcal{F}_S \in \mathbb{R}^{H_S \times W_S \times D}$, the initial static template feature map $\mathcal{F}_{T_S} \in \mathbb{R}^{H_{TS} \times W_{TS} \times D}$, and the dynamic template feature map $\mathcal{F}_{T_D} \in \mathbb{R}^{H_{TD} \times W_{TD} \times D}$. Then flattened and concatenated along the channel dimension to form a joint feature map $\boldsymbol{X}$, as shown in the following equation:

$$\boldsymbol{X} = \mathbf{Concat}\left(\mathbf{Flatten}\left(\mathcal{F}_S\right), \ \mathbf{Flatten}\left(\mathcal{F}_{T_S}\right), \ \mathbf{Flatten}\left(\mathcal{F}_{T_D}\right)\right), \tag{3.1}$$

where **Concat** denotes the concatenation operator and **Flatten** denotes the flattening operator. Specifically, the feature maps are reshaped using a $r \times r$ size window, then through concatenation resulting in $L = \left(\frac{H_S}{r} \times \frac{W_S}{r} + \frac{H_{T_S}}{r} \times \frac{W_{T_S}}{r} + \frac{H_{T_D}}{r} \times \frac{W_{T_D}}{r}\right)$ tokens.

Positional encoding $\boldsymbol{P}_E$ is added to the joint feature map $\boldsymbol{X}$ to incorporate spatial information, forming the feature fusion block input $\boldsymbol{Z}_0$, as shown in the following equation:

$$\boldsymbol{Z}_0 = \boldsymbol{X} + \boldsymbol{P}_E. \tag{3.2}$$

The positional encoding $\boldsymbol{P}_E$ is defined as follows:

$$\boldsymbol{P}_E(n,i)=\begin{cases}\mathbf{sin}\left(\dfrac{n}{10000^{\frac{i}{d_m}}}\right), if\ i=2k\\ \mathbf{cos}\left(\dfrac{n}{10000^{\frac{i}{d_m}}}\right), if\ i=2k+1\end{cases}, \tag{3.3}$$

where n represents the position of the token, i represents the index of the element within the token, and d_m is the dimension of the positional encoding vector, with a size of r^2D, $\boldsymbol{P}_E\in\mathbb{R}^{L\times(r^2D)}$.

3.2. Feature fusion block

In this section, we introduce the feature fusion block (FFB). As mentioned in the preliminaries, transferring spatial-domain computation to the frequency-domain enhances computational efficiency and reduces memory usage. The FFB has two key parts: the spatial-domain feature fusion module (SFM) and the frequency-domain feature fusion module (FFM). These two modules integrate spatial and frequency domain features, improving the model's spatial context awareness. The following sections provide detailed explanations of these two modules.

3.2.1. Spatial-domain feature fusion module (SFM)

The FFB input $\boldsymbol{Z}_j$ is divided into eight feature heads, as shown in Eq (3.4)

$$[\mathcal{X}_0,\cdots,\mathcal{X}_8]=\mathbf{Split}\left(\mathbf{Reshape}\left(\boldsymbol{Z}_j\right)\right), \tag{3.4}$$

where $\mathbf{Split}$ denotes the splitting operator, and the size of each head after splitting is $\mathcal{X}_i\in\mathbb{R}^{L\times r\times r\times d_i}$, $i=1,2,\cdots,8$.

Each feature head undergoes linear transformations to generate the query, key, and value feature tensors, i.e., $\mathcal{Q}$, $\mathcal{K}$ and $\mathcal{V}$; the linear transformation process is given by Eq (3.5)

$$\begin{cases}\mathcal{Q}&=\mathbf{LP}_1(\mathcal{X}_i)\\ \mathcal{K}&=\mathbf{LP}_2(\mathcal{X}_i),\\ \mathcal{V}&=\mathbf{LP}_3(\mathcal{X}_i)\end{cases} \tag{3.5}$$

where $\mathbf{LP}$ denotes linear mapping transformation. The sizes of feature maps are $\mathcal{Q}\in\mathbb{R}^{L\times r\times r\times d_i}$, $\mathcal{K}\in\mathbb{R}^{L\times r\times r\times d_i}$, and $\mathcal{V}\in\mathbb{R}^{L\times r\times r\times d_i}$. The feature tensor $\mathcal{Q}$ is reshaped into a query matrix $\boldsymbol{Q}$, with a size of $\boldsymbol{Q}\in\mathbb{R}^{L\times(r^2d_i)}$.

The key and value feature tensors $\mathcal{K}$, $\mathcal{V}$ are subjected to CS operations to generate the shifted key tensor $\mathcal{K}_C$ and the shifted value tensor $\mathcal{V}_C$, respectively, as shown in Eq (3.6)

$$\begin{cases} \mathcal{K}_C = \mathbf{CS}(\mathcal{K}) \\ \mathcal{V}_C = \mathbf{CS}(\mathcal{V}) \end{cases}, \tag{3.6}$$

where $\mathbf{CS}$ performs a cyclic-shift along the second and third dimensions of the input tensor. The size of the shifted key tensor $\mathcal{K}_C$ and the value tensor $\mathcal{V}_C$ are $\mathcal{K}_C \in \mathbb{R}^{L\times r^2\times r\times r\times d_i}$ and $\mathcal{V}_C \in \mathbb{R}^{L\times r^2\times r\times r\times d_i}$.

The shifted key tensor and shifted value tensor are then reshaped to obtain the shifted key matrix $\boldsymbol{K}_C$ and the shifted value matrix $\boldsymbol{V}_C$, respectively

$$\begin{cases} \boldsymbol{K}_C = \mathbf{Reshape}(\mathcal{K}_C) \\ \boldsymbol{V}_C = \mathbf{Reshape}(\mathcal{V}_C) \end{cases}, \tag{3.7}$$

where $\boldsymbol{K}_C \in \mathbb{R}^{(Lr^2)\times(r^2 d_i)}$ and $\boldsymbol{V}_C \in \mathbb{R}^{(Lr^2)\times(r^2 d_i)}$.

After reshaping, the shifted key matrix $\boldsymbol{K}_C$ is transposed to obtain the transposed key matrix $\boldsymbol{K}_C^{\mathrm{T}} \in \mathbb{R}^{(r^2 d_i)\times(Lr^2)}$. The transposed key matrix $\boldsymbol{K}_C^{\mathrm{T}}$ is then scaled by a factor $\sqrt{r^2 d_i}$ to facilitate the normalization operation of the Softmax function and avoid gradient vanishing or explosion. The query matrix $\boldsymbol{Q}$ and the scaled key matrix $\boldsymbol{K}_{CN} = \dfrac{\boldsymbol{K}_C^{\mathrm{T}}}{\sqrt{r^2 d_i}}$ are multiplied to obtain the feature attention matrix $\boldsymbol{A}$, as shown in Eq (3.8)

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}_{CN}, \tag{3.8}$$

where $\boldsymbol{A} \in \mathbb{R}^{L\times(Lr^2)}$.

During the CS process, the concatenation of different regions does not guarantee the continuity of the foreground target, leading to discontinuities at the boundaries of the concatenated images, known as boundary effects. To mitigate these effects caused by CS, an attention mask matrix $\boldsymbol{M}$ is introduced to penalize shifted samples that are far from the base sample. The feature attention score matrix is added to the attention mask matrix to obtain the final feature attention matrix $\boldsymbol{A}_M$, as shown in Eq (3.9)

$$\begin{cases} \boldsymbol{A}_M = \boldsymbol{A} + \boldsymbol{M} \\ \boldsymbol{M} = \mathbf{1}_{L\times L} \otimes \left(\boldsymbol{vec}(\boldsymbol{C})\right)^{\mathrm{T}} \\ \boldsymbol{C} = \boldsymbol{y}^{\mathrm{T}} \times \boldsymbol{y} - \mathbf{1}_{r\times r} \end{cases}, \tag{3.9}$$

where $\mathbf{1}_{L\times L}$ is a matrix of size $L \times L$ with all elements equal to 1. In the same way, $\mathbf{1}_{r\times r}$ is a matrix of size $r\times r$ with all elements equal to 1. The symbol $\otimes$ denotes the Kronecker product, with $\boldsymbol{y} \in \mathbb{R}^{1\times r}$, $y_i = \begin{cases} \boldsymbol{x}_i, \boldsymbol{x}_i > 0.5 \\ 1 - \boldsymbol{x}_i, others \end{cases}$, $i = 1, 2, \cdots, r$, $\boldsymbol{x} \in \mathbb{R}^{1\times r}$, and $\boldsymbol{x}_i = \dfrac{i-1}{r}$.

The Softmax function transforms raw attention scores into a probability distribution. The normalized attention matrix is then multiplied by the shifted value matrix $\boldsymbol{V}_C$ to generate the single-head attention output $\boldsymbol{T}_i$

$$\boldsymbol{T}_i = \textbf{Softmax}\left(\boldsymbol{A}_M\right)\times\boldsymbol{V}_C, \tag{3.10}$$

where $\boldsymbol{T}_i \in \mathbb{R}^{L\times(r^2 d_i)}$ $(i = 1, 2, \cdots, 8)$.

The single-head attention matrix is passed through a feed-forward neural network (FFN) to enhance the features. Specifically, the multi-head attention matrix $\boldsymbol{O}$ is formed by concatenating all single-head outputs, as shown in Eq (3.11)

$$\boldsymbol{O} = \textbf{Concat}\left(\textbf{LP}_1\left(\boldsymbol{T}_1\right), \cdots, \textbf{LP}_8\left(\boldsymbol{T}_8\right)\right), \tag{3.11}$$

where $\boldsymbol{O} \in \mathbb{R}^{\left(Lr^2\right)\times D}$.

The multi-head attention matrix $\boldsymbol{O}$ is added to the FFB input $\boldsymbol{Z}_j$ and then normalized using layer normalization, as shown in Eq (3.12)

$$\boldsymbol{T}_{temp} = \textbf{LN}\left(\textbf{LP}\left(\boldsymbol{O}\right) + \boldsymbol{Z}_j\right). \tag{3.12}$$

$\boldsymbol{T}_{temp}$ is passed through the FFN for non-linear transformation, and the result is added back to the original $\boldsymbol{T}_{temp}$ matrix to achieve residual connection and normalization. This process aims to preserve as much of the underlying structural information of the input data as possible. By integrating the deep semantic features with the original spatial details in an organic manner, a composite representation that combines high fidelity and strong discriminability is ultimately generated, as shown in Eq (3.13)

$$\boldsymbol{Z}_{j+1} = \textbf{LN}\left(\textbf{FFN}\left(\boldsymbol{T}_{temp}\right) + \boldsymbol{T}_{temp}\right), \tag{3.13}$$

where the FFN consists of two linear mapping layers and a non-linear rectified linear unit(ReLU) activation layer, as shown in Eq (3.14)

$$\textbf{FFN}\left(\boldsymbol{X}\right) = \left(\textbf{ReLU}\left(\boldsymbol{X}\boldsymbol{L}_1 + \boldsymbol{B}_1\right)\right)\boldsymbol{L}_2 + \boldsymbol{B}_2, \tag{3.14}$$

where $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ are the weight matrices for linear projection, and $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ are the bias matrices.

Finally, the FFB re-inputs the obtained multi-head attention matrices. Each layer's output serves as the input for the next layer. After repeating this process six times, six channels of attention computation results are obtained. This process yields a more accurate final attention result, as shown in Eq (3.15)

$$\boldsymbol{Z}_{j+1} = \textbf{FFB}\left(\boldsymbol{Z}_j\right), \tag{3.15}$$

where $j = 0, 1, \cdots, 5$.

3.2.2. Frequency-domain feature fusion module (FFM)

Given that the data generated by circulant matrices are redundant, direct computation in the spatial domain results in high computational complexity and extensive memory usage. Consequently, transforming circulant matrix operations from spatial domain calculations to frequency domain element-wise multiplications is feasible for reducing the computational overhead. When the window size is 8, the FFM is involved in network training and inference. The structure of the FFM is shown in Figure 2.

Through Fourier transforms, the query feature tensor $\mathcal{Q}$ and the key feature tensor $\mathcal{K}$, obtained from spatial domain computations, are converted from the spatial domain to the frequency domain, as shown in Eq (3.16)

$$\begin{cases} \widehat{\mathcal{Q}} = \mathbf{fft}_{[2,\,3]}\left(\mathcal{Q}\right) \\ \widehat{\mathcal{K}} = \mathbf{fft}_{[2,\,3]}\left(\mathcal{K}\right) \end{cases}, \tag{3.16}$$

where $\mathcal{Q} \in \mathbb{R}^{L\times r\times r\times d_i}$, $\mathcal{K} \in \mathbb{R}^{L\times r\times r\times d_i}$, and $\mathbf{fft}_{[2,\,3]}$ denotes the Fourier transform operation applied to the second and third dimensions of the feature tensors $\mathcal{Q}$ and $\mathcal{K}$, resulting in $\widehat{\mathcal{Q}} \in \mathbb{C}^{L\times r\times r\times d_i}$ and $\widehat{\mathcal{K}} \in \mathbb{C}^{L\times r\times r\times d_i}$.

The query feature tensor in the frequency domain $\widehat{\mathcal{Q}}$ is then expanded, with each token being copied L times to obtain the expanded query feature tensor $\widehat{\mathcal{Q}}_{kE}$, as shown in the following equation:

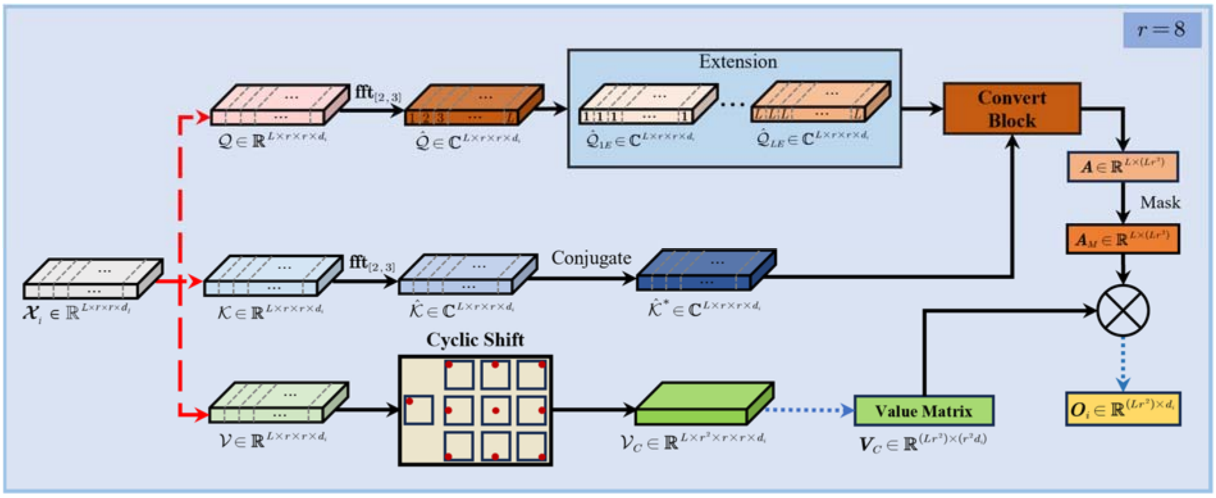
$$\begin{cases} \widehat{\mathcal{Q}}_{1E} = \mathbf{Repeat}\left(\widehat{\mathcal{Q}}\left(1,\,:,\,:,:\,\right),\,L\right) \\ \widehat{\mathcal{Q}}_{2E} = \mathbf{Repeat}\left(\widehat{\mathcal{Q}}\left(2,\,:,\,:,\,:\right),\,L\right) \\ \vdots \\ \widehat{\mathcal{Q}}_{LE} = \mathbf{Repeat}\left(\widehat{\mathcal{Q}}\left(L,\,:,\,:,\,:\right),\,L\right) \end{cases}, \tag{3.17}$$

where **Repeat** denotes the repetition of each input tensor L times along the first dimension, with $\widehat{\mathcal{Q}}_{kE} \in \mathbb{C}^{L\times r\times r\times d_i}$, $k = 1,\,2,\,\cdots,L$, $\widehat{\mathcal{Q}}\left(l,\,:,\,:,\,:\right) \in \mathbb{C}^{1\times r\times r\times d_i}$, and $l = 1,\,2,\,\cdots,L$.

The conjugate of the key feature tensor in the frequency domain $\widehat{\mathcal{K}}^{*}$ is computed, as shown in the following equation:

$$\widehat{\mathcal{K}}^{*} = \mathbf{Conjugate}\left(\widehat{\mathcal{K}}\right), \tag{3.18}$$

where **Conjugate** is the conjugation operator that negates the imaginary part of the input tensor.



$r=8$
Extension
Convert Block
$\mathrm{fft}_{[2,3]}$
$\mathcal{Q}\in\mathbb{R}^{L\times r\times r\times d_i}$
$\hat{\mathcal{Q}}\in\mathbb{C}^{L\times r\times r\times d_i}$
$\hat{\mathcal{Q}}_{1E}\in\mathbb{C}^{L\times r\times r\times d_i}$
$\hat{\mathcal{Q}}_{LE}\in\mathbb{C}^{L\times r\times r\times d_i}$
$\boldsymbol{A}\in\mathbb{R}^{L\times(Lr^2)}$
Mask
$\boldsymbol{A}_M\in\mathbb{R}^{L\times(Lr^2)}$
$\mathcal{X}_i\in\mathbb{R}^{L\times r\times r\times d_i}$
$\mathcal{K}\in\mathbb{R}^{L\times r\times r\times d_i}$
$\mathrm{fft}_{[2,3]}$
$\hat{\mathcal{K}}\in\mathbb{C}^{L\times r\times r\times d_i}$
Conjugate
$\hat{\mathcal{K}}^*\in\mathbb{C}^{L\times r\times r\times d_i}$
Cyclic Shift
$\mathcal{V}\in\mathbb{R}^{L\times r\times r\times d_i}$
$\mathcal{V}_C\in\mathbb{R}^{L\times r^2\times r\times r\times d_i}$
Value Matrix
$\boldsymbol{V}_C\in\mathbb{R}^{(Lr^2)\times(r^2d_i)}$
$\boldsymbol{O}_i\in\mathbb{R}^{(Lr^2)\times d_i}$

Figure 2. Diagram of the FFM architecture.

To simplify the complex spatial domain computation, the convert block executes element-wise conversion between the expanded feature tensor $\widehat{\mathcal{Q}}_{kE}$ and the conjugate feature tensor $\widehat{\mathcal{K}}^*$ obtained above. The results from different channels of the resulting tensor are then summed. The summed result in the frequency domain needs to be transformed back to the spatial domain. Here, the inverse Fourier transform is applied to the frequency domain result, and the real part of the transformed result is taken and stacked as the l-th row of the attention weight matrix $\boldsymbol{A}$, as shown in the following equation:

$$\begin{aligned}\boldsymbol{A}(k,:)&=\frac{1}{\sqrt{r^2 d_i}}\mathbf{Concat}\left\{\mathbf{vec}\left[\sum_{c=1}^{d_i}\mathcal{Q}(k,:,:,c)\star\mathcal{K}(p,:,:,c)\right]^{\mathrm{T}}\right\}\\&=\left(\mathbf{vec}\left(\mathbf{real}\left(\mathbf{ifft}_2\left(\frac{\sum_{c=1}^{d_i}\widehat{\mathcal{Q}}_{kE}(:,:,:,\mathrm{c})\odot\widehat{\mathcal{K}}^*(:,:,:,\mathrm{c})}{\sqrt{r^2 d_i}}\right)\right)\right)\right)^{\mathrm{T}},\end{aligned}\tag{3.19}$$

where **vec** denotes the vectorization operator for tensors, **real** denotes the operation of taking the real part of the tensor, $\mathbf{ifft}_2$ is the two-dimensional inverse Fourier transform, the symbol $\odot$ represents element-wise multiplication, and the symbol $\star$ denotes the correlation operation between matrices. For better comprehension of the convert block, the subsequent text further explains Eq (3.19).

The query feature map tensor $\mathcal{Q}$ and the key feature map tensor $\mathcal{K}$ both contain L tokens of size $r\times r\times d$. After the CS operation, as shown in Eq (3.6), the shifted key tensor $\mathcal{K}_C$ is obtained. The query feature map tensor $\mathcal{Q}$ is reshaped into the query matrix $\boldsymbol{Q}$, and the shifted key tensor $\mathcal{K}_C$ is also reshaped into the shifted key matrix $\boldsymbol{K}_C$. The transpose of the shifted key matrix $\boldsymbol{K}_C^{\mathrm{T}}$ is then computed and multiplied with the query matrix $\boldsymbol{Q}$ to get the feature attention matrix $\boldsymbol{A}$, as shown in the following equation:

$$\boldsymbol{A}(k,n)=\frac{\boldsymbol{Q}(k,:)\times\boldsymbol{K}_C^{\mathrm{T}}(:,n)}{\sqrt{r^2 d_i}}=\frac{\left\langle\mathcal{Q}(k,:,:,:),\mathcal{K}_C(p,q,:,:,:)\right\rangle}{\sqrt{r^2 d_i}},\tag{3.20}$$

where $\boldsymbol{Q}(k,:)$ denotes the k-th row of $\boldsymbol{Q}$, with a size of $\boldsymbol{Q}(k,:)\in\mathbb{R}^{r^2\times d_i}$; $\boldsymbol{K}_C^{\mathrm{T}}(:,n)$ denotes any column in $\boldsymbol{K}_C^{\mathrm{T}}$, with each column having a size of $\boldsymbol{K}_C^{\mathrm{T}}(:,n)\in\mathbb{R}^{r^2\times d_i}$, where $n=(q-1)L+p$; $\boldsymbol{A}(k,n)$ represents the element in the k-th row and n-th position of $\boldsymbol{A}$. $\langle\mathcal{Q}(k,:,:,:),\mathcal{K}_C(p,q,:,:,:)\rangle$ represents the inner product between the query feature map tensor $\mathcal{Q}$ and the shifted key tensor $\mathcal{K}_C$. $\mathcal{Q}(k,:,:,:)$ denotes the k-th token of $\mathcal{Q}$, and $\mathcal{K}_C(p,q,:,:,:)$ denotes the q-th token in the p-th group of $\mathcal{K}_C$, where $p=1,2,\cdots,L$ and $q=1,2,\cdots,r^2$.

As indicated in the preliminaries, matrix multiplication is convertible to tensor correlation operations, as shown in Eq (3.21)

$$\begin{aligned}&\left[\boldsymbol{A}(k,(1-1)L+p),\boldsymbol{A}(k,(2-1)L+p),\cdots,\boldsymbol{A}\left(k,\left(r^2-1\right)L+p\right)\right]\\&=\frac{1}{\sqrt{r^2d_i}}\mathbf{vec}\left[\sum_{c=1}^{d_i}\mathcal{Q}(k,:,:,c)\star\mathcal{K}_C(p,1,:,:,c)\right]^{\mathrm{T}}\\&=\frac{1}{\sqrt{r^2d_i}}\mathbf{vec}\left[\sum_{c=1}^{d_i}\mathcal{Q}(k,:,:,c)\star\mathcal{K}(p,:,:,c)\right]^{\mathrm{T}},\end{aligned}\tag{3.21}$$

where $\mathbf{vec}$ denotes the vectorization operator.

The correlation operation is viewed as a convolution operation between tensors, as shown in Eq (3.22).

$$\mathcal{Q}(k,:,:,c)\star\mathcal{K}_C(p,1,:,:,c)=\mathcal{Q}(k,:,:,c)*\overline{\mathcal{K}}_C(p,1,:,:,c),\tag{3.22}$$

where $\overline{\mathcal{K}}_C(p,1,:,:,c)$ denotes the flipped version of $\mathcal{K}_C$, $p=1,2,\cdots,L$, and $q=1,2,\cdots,r^2$.

Thus, Eq (3.21) is rewritten as Eq (3.23)

$$\begin{aligned}&\left[\boldsymbol{A}(k,(1-1)L+p),\boldsymbol{A}(k,(2-1)L+p),\cdots,\boldsymbol{A}\left(k,\left(r^2-1\right)L+p\right)\right]\\&=\frac{1}{\sqrt{r^2d_i}}\mathbf{vec}\left[\sum_{c=1}^{d_i}\mathcal{Q}(k,:,:,c)*\overline{\mathcal{K}}_C(p,1,:,:,c)\right]^{\mathrm{T}}.\end{aligned}\tag{3.23}$$

By traversing p and concatenating into a new row vector, $n=(q-1)L+p$ is also traversed, as shown in the following equation:

$$\boldsymbol{A}(k,:)=\frac{1}{\sqrt{r^2d_i}}\mathbf{Concat}\left\{\mathbf{vec}\left[\sum_{c=1}^{d_i}\mathcal{Q}(k,:,:,c)*\overline{\mathcal{K}}_C(p,:,:,c)\right]^{\mathrm{T}}\right\}.\tag{3.24}$$

According to the convolution theorem, the correlation operation in the spatial domain is interchangeable with element-wise multiplication in the frequency domain. Hence, Eq (3.19) is proven. The proposed FFB algorithm in this paper is shown in Algorithm 1.

Algorithm 1: Feature fusion block (FFB)

Input: Input features $\boldsymbol{Z}_0$, the number of heads in the FFB I, and the vector $\boldsymbol{r} = [1, 2, 4, 8, 1, 2, 4, 8]$ constructed from window sizes of each attention head and number of layers in the FFB J.

Output: Output features $\boldsymbol{Z}_6$.

```
1:  Split $\boldsymbol{Z}_0$ into $I$ heads.
2:  For $i \leftarrow 0$ to $I-1$
3:      For $j \leftarrow 0$ to $J-1$
4:          If $r_{i+1} = 1$ or $r_{i+1} = 2$ or $r_{i+1} = 4$
5:              Let $\boldsymbol{Z}_j$ as the input to the SFM in Figure 1, calculate the attention weight matrix $\boldsymbol{A}$ via Eqs (3.6)–(3.9).
6:          Else
7:              Let $\boldsymbol{Z}_j$ as the input to the FFM in Figure 2, calculate the attention weight matrix $\boldsymbol{A}$ via Eqs (3.16)–(3.19).
8:          End if
9:      End for
10: End for
11: The fused features are further calculated using Eqs (3.11)–(3.14).
12: Return $\boldsymbol{Z}_6$.
```

3.3. Feature enhancement block

The feature enhancement block (FEB) and FFB have an identical number of stacked layers, with both having six layers, each containing eight attention heads. The main structure of the FEB is made up of multi-head self-attention (MSA) layers, multi-head cross-attention (MCA) layers, and feed-forward networks (FFN). This part draws on the concept from the DETR [53] object detection model, which employs the learnability of queries to steer object detection. Figure 3 displays the signal flow diagram of FEB, where the query embedding $\boldsymbol{Q}_E$ is a position encoding learned in the model training process, aiding the model in obtaining the spatial location information of the object.

First, the FEB input is initialized with zero vectors, i.e. $\boldsymbol{d}_0 = \boldsymbol{Query} = \boldsymbol{0} \in \mathbb{R}^{1\times D}$, and added to the query embedding vector $\boldsymbol{Q}_E$ to form the query matrix $\boldsymbol{Q}_s$ and the key matrix $\boldsymbol{K}_s$, which carry the object's position information. The value matrix $\boldsymbol{V}_s$ is directly provided by the object sequence $\boldsymbol{Query}$, as shown in the following equation:

$$\begin{cases} \left[\boldsymbol{Q}_{s_1}, \boldsymbol{Q}_{s_2}, \cdots, \boldsymbol{Q}_{s_8}\right] = \textbf{Split}\left[\boldsymbol{Q}_s \boldsymbol{W}_{Q_s}\right] \\ \left[\boldsymbol{K}_{s_1}, \boldsymbol{K}_{s_2}, \cdots, \boldsymbol{K}_{s_8}\right] = \textbf{Split}\left[\boldsymbol{K}_s \boldsymbol{W}_{K_s}\right]. \\ \left[\boldsymbol{V}_{s_1}, \boldsymbol{V}_{s_2}, \cdots, \boldsymbol{V}_{s_8}\right] = \textbf{Split}\left[\boldsymbol{V}_s \boldsymbol{W}_{V_s}\right] \end{cases} \tag{3.25}$$

The obtained $\boldsymbol{Q}_s$, $\boldsymbol{K}_s$, and $\boldsymbol{V}_s$ are fed into the MSA layer to compute the similarity between $\boldsymbol{Q}_s$ and $\boldsymbol{K}_s$ to identify the features most relevant to the object. The output of the MSA layer $\boldsymbol{s}$ is iterated, as shown in the following equation:

$$\begin{cases} z_{s_i} = \text{Attention}\left(Q_{s_i}, K_{s_i}, V_{s_i}\right) = \textbf{Softmax}\left(\dfrac{Q_{s_i} K_{s_i}^{\mathrm{T}}}{\sqrt{d_i}}\right) V_{s_i} \\ s = \textbf{MSA}\left(Q_s, K_s, V_s\right) = \textbf{Concat}\left(z_{s_1}, z_{s_2}, \cdots, z_{s_8}\right) W_s \end{cases}, \tag{3.26}$$

where W_{Q_s}, W_{K_s}, W_{V_s}, and W_s are linear projection weight matrices learned through the network, $s \in \mathbb{R}^{1\times D}$.

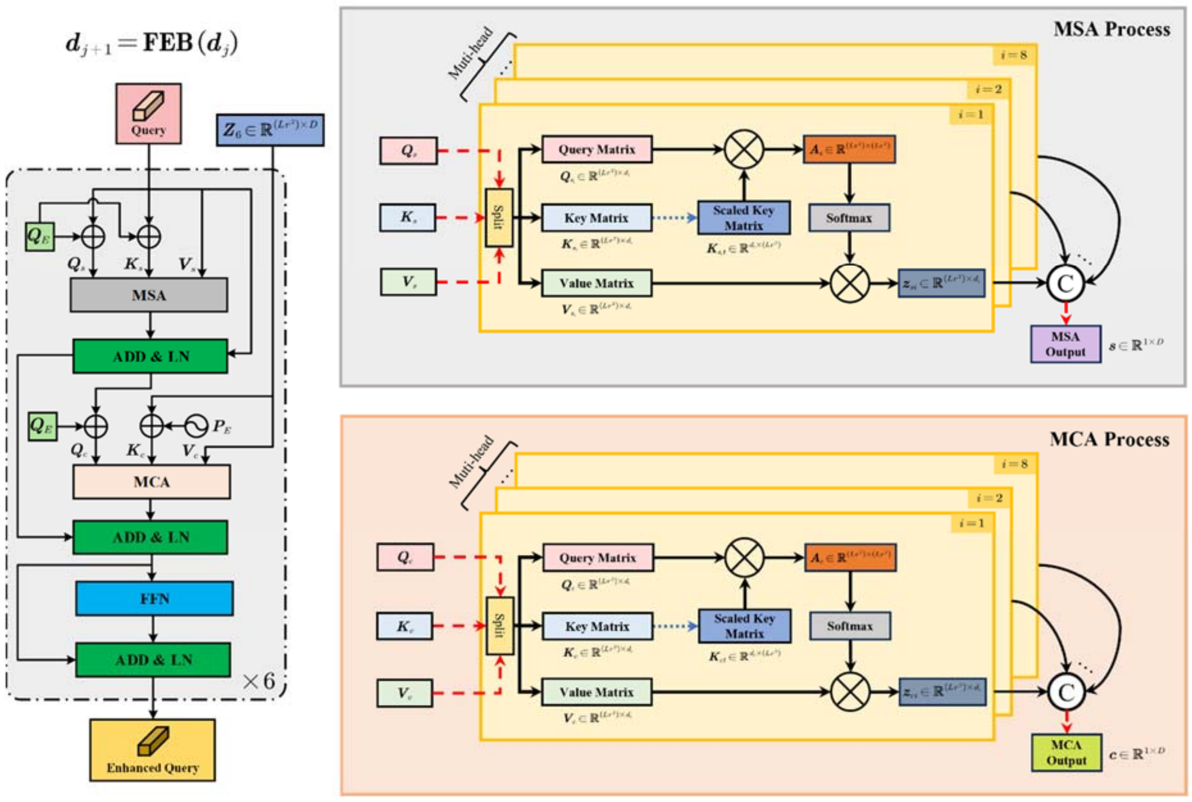


Figure 3. The signal flow diagram of the FEB.

To avoid the occurrence of gradient explosion or vanishing, the output of the MSA layer s is subjected to residual connection and followed by a normalization layer to obtain the intermediate state output, as shown in the following equation:

$$h_1 = \textbf{LN}\left(\textbf{LP}(s) + d_j\right). \tag{3.27}$$

The query embedding vector Q_E is added to the intermediate state output h_1, resulting in the query matrix Q_c for the MCA layer. The final output of the feature fusion block Z_6 is combined with the position encoding vector P_E, forming the key matrix for the MCA layer K_c; Z_6 also serves as the value matrix in the MCA layer calculation, as illustrated in Eq (3.28).

$$\begin{cases} \boldsymbol{Q}_c = \boldsymbol{h}_1 + \boldsymbol{Q}_E \\ \boldsymbol{K}_c = \boldsymbol{Z}_6 + \boldsymbol{P}_E \\ \boldsymbol{V}_c = \boldsymbol{Z}_6 \end{cases}. \tag{3.28}$$

Similarly $\boldsymbol{Q}_c$, $\boldsymbol{K}_c$, and $\boldsymbol{V}_c$ are processed as shown in the following equation:

$$\begin{cases} \left[\boldsymbol{Q}_{c_1}, \boldsymbol{Q}_{c_2}, \cdots, \boldsymbol{Q}_{c_8}\right] = \mathbf{Split}\left[\boldsymbol{Q}_c \boldsymbol{W}_{Q_c}\right] \\ \left[\boldsymbol{K}_{c_1}, \boldsymbol{K}_{c_2}, \cdots, \boldsymbol{K}_{c_8}\right] = \mathbf{Split}\left[\boldsymbol{K}_c \boldsymbol{W}_{K_c}\right] \\ \left[\boldsymbol{V}_{c_1}, \boldsymbol{V}_{c_2}, \cdots, \boldsymbol{V}_{c_8}\right] = \mathbf{Split}\left[\boldsymbol{V}_c \boldsymbol{W}_{V_c}\right] \end{cases}. \tag{3.29}$$

The calculation process of the MCA layer is as follows:

$$\begin{cases} \boldsymbol{z}_{c_i} = \text{Attention}\left(\boldsymbol{Q}_{c_i}, \boldsymbol{K}_{c_i}, \boldsymbol{V}_{c_i}\right) = \mathbf{Softmax}\left(\frac{\boldsymbol{Q}_{c_i} \boldsymbol{K}_{c_i}^{\mathrm{T}}}{\sqrt{d_i}}\right) \boldsymbol{V}_{c_i} \\ \boldsymbol{c} = \mathbf{MCA}\left(\boldsymbol{Q}_c, \boldsymbol{K}_c, \boldsymbol{V}_c\right) = \mathbf{Concat}\left(\boldsymbol{z}_{c_1}, \boldsymbol{z}_{c_2}, \cdots, \boldsymbol{z}_{c_8}\right) \boldsymbol{W}_c \end{cases}, \tag{3.30}$$

where $\boldsymbol{W}_{Q_c}$, $\boldsymbol{W}_{K_c}$, $\boldsymbol{W}_{V_c}$, and $\boldsymbol{W}_c$ are the linear projection weight matrices learned through the network, with $\boldsymbol{c} \in \mathbb{R}^{1 \times D}$.

The output of the MCA layer $\boldsymbol{c}$ is subjected to residual connection and normalization to obtain the intermediate output state $\boldsymbol{h}_2$, as shown in the following equation:

$$\boldsymbol{h}_2 = \mathbf{LN}\left(\mathbf{LP}\left(\boldsymbol{c}\right) + \boldsymbol{h}_1\right). \tag{3.31}$$

The process described above is repeated six times, constituting the entire FEB workflow. The final output of the FFB is then fed into the prediction analysis module for subsequent tracking and prediction. The overall process is represented by Eq (3.32)

$$\boldsymbol{d}_{j+1} = \mathbf{FEB}\left(\boldsymbol{d}_j\right), \tag{3.32}$$

where $j = 0, 1, \cdots, 5$.

The FEB, together with the FFB, forms the encoder-decoder structure of the Transformer, which helps the model build temporal context awareness and obtain richer spatiotemporal contextual information, thereby further enhancing the precision of the model.

3.4. Prediction analysis module

The prediction analysis module includes a bounding box prediction head and a classification score head. The enhanced feature $\boldsymbol{d}_6$ is directed into these heads for analysis, thus accomplishing further tracking and prediction.

3.4.1. Bounding box prediction head

The bounding box prediction head is designed inspired by the corner prediction head in the STARK [43] algorithm. Calculating the expected values of the probability distributions of the two probability maps corresponding to the top-left and bottom-right corners of the tracked object's bounding box, thus obtaining the coordinates of these corners, which is illustrated in Figure 4.

First, the final output of the feature enhancement module $\boldsymbol{Z}_6$ is cropped to obtain the search region features $\boldsymbol{Z}_S \in \mathbb{R}^{(H_S W_S)\times D}$. The enhanced feature $\boldsymbol{d}_6$ is also input to calculate the similarity modulation vector $\boldsymbol{m}$ between them, as shown in the following equation:

$$\boldsymbol{m} = \boldsymbol{Z}_S \boldsymbol{d}_6^{\mathrm{T}}, \tag{3.33}$$

where $\boldsymbol{m} \in \mathbb{R}^{(H_S W_S)\times 1}$.

To better enhance the important tracking regions and weaken the interference of other regions on the tracking, the similarity modulation vector $\boldsymbol{m}$ is combined with the search region features $\boldsymbol{Z}_S$ via element-wise multiplication, yielding the enhanced search region features $\boldsymbol{Z}_E$, as shown in the following equation:

$$\boldsymbol{Z}_E = \boldsymbol{Z}_S \odot \mathbf{Repeat}(\boldsymbol{m}, D). \tag{3.34}$$

Among them, $\mathbf{Repeat}(\boldsymbol{m}, D) \in \mathbb{R}^{(H_S W_S)\times D}$ indicates that the column vector $\boldsymbol{m}$ is continuously copied D times and stacked along the row direction to generate a matrix.

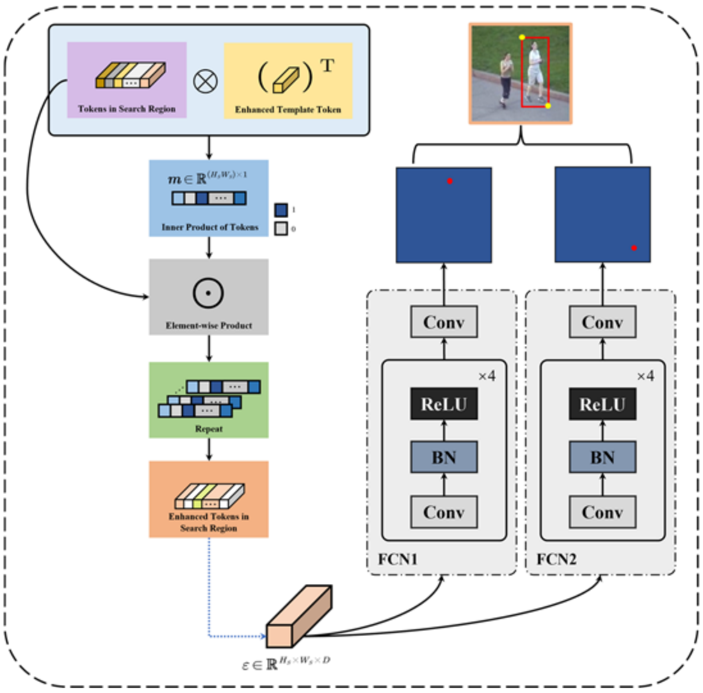


Figure 4. The flowchart of the bounding box prediction head.

The obtained enhanced search region features $\boldsymbol{Z}_E$ are reshaped into an enhanced feature map $\boldsymbol{\varepsilon} \in \mathbb{R}^{H_S \times W_S \times D}$, which undergoes processing through a fully convolutional network (FCN), resulting in two probability distribution maps for the bounding box's top-left and bottom-right corners, namely $\boldsymbol{P}_1(x, y)$ and $\boldsymbol{P}_2(x, y)$, as shown in the following equation:

$$\begin{cases} \boldsymbol{P}_1(x, y) = \mathbf{FCN}_1(\boldsymbol{\varepsilon}) \\ \boldsymbol{P}_2(x, y) = \mathbf{FCN}_2(\boldsymbol{\varepsilon}) \end{cases}, \tag{3.35}$$

where the FCN consists of five convolutional layers, batch normalization layers, and ReLU activation layers (Conv-BN-ReLU). $\mathbf{FCN}_i$ represents the processing of the data by two FCN layers with different parameters.

To determine the coordinates of the bounding box's two corner points, the two probability maps $\boldsymbol{P}_1(x, y)$ and $\boldsymbol{P}_2(x, y)$ are utilized through the following equation. After acquiring these coordinates, the final predicted bounding box position candidate samples $\mathcal{C}_p \in \mathbb{R}^{H_C \times W_C \times 3}$ are determined.

$$\begin{cases} (\hat{x}_1, \hat{y}_1) = \left(\sum_{y=0}^{H} \sum_{x=0}^{W} x\boldsymbol{P}_1(x, y),\ \sum_{y=0}^{H} \sum_{x=0}^{W} y\boldsymbol{P}_1(x, y) \right) \\ (\hat{x}_2, \hat{y}_2) = \left(\sum_{y=0}^{H} \sum_{x=0}^{W} x\boldsymbol{P}_2(x, y),\ \sum_{y=0}^{H} \sum_{x=0}^{W} y\boldsymbol{P}_2(x, y) \right) \end{cases}, \tag{3.36}$$

where $(\hat{x}_1, \hat{y}_1)$ represents the coordinates of the top-left corner and similarly $(\hat{x}_2, \hat{y}_2)$ stands for the coordinates of the bottom-right corner. Suppose that the prediction score of the candidate sample exceeds the set threshold. In that case, the template area is expanded to twice its original size, with the center of the sample as the base point, forming a dynamic template and updating it.

3.4.2. Classification score head

During the object tracking process, the object may undergo varying changes as the tracking time progresses. Therefore, real-time capture of its latest state is crucial for accurate tracking. As shown in Figure 1, we propose a dynamic template update mechanism sampled from intermediate frames. It serves as an additional input to capture the object's changes in appearance over time and offers more temporal details. However, when the object is entirely occluded or moves out of view, the update of the dynamic template may become unreliable. Therefore, we designed a simple candidate sample scoring head to assess whether the current sample needs to be updated.

First, the enhanced feature $\boldsymbol{d}_6$ is processed through a multilayer perceptron to improve the model's learning ability. Then the output results are activated by a function to derive the score, as shown in the following equation:

$$s = \mathbf{Sigmoid}\left(\mathbf{MLP}(\boldsymbol{d}_6)\right), \tag{3.37}$$

where $\mathbf{MLP}$ is a three-layer perceptron with ReLU activation functions employed in all hidden layers.

The output layer employs the function $\mathbf{Sigmoid}(x)=\frac{1}{1+e^{-x}}$ as the activation function to map the output values to the 0 to 1 range. As shown in Figure 1, when the score s exceeds the set threshold, the model considers the candidate sample to be reliable and updates the dynamic template; otherwise, it does not update. Through this mechanism, the model has more effectively judged the reliability of candidate samples in complex environments, thereby improving the stability and accuracy of tracking.

3.4.3. Loss function

We draw on the loss function design concept in the DETR architecture, adopting an end-to-end training method and using the generalized intersection over union (GIOU) loss to optimize the predicted bounding boxes. By directly optimizing the intersection over union (IOU) between the predicted boxes and the ground-truth boxes, the accuracy and robustness of the tracking are enhanced. In addition, L_1 loss is introduced to optimize the position and size of the bounding box, enabling the model to predict the object's position more accurately. The loss function is shown in the following equation:

$$L=\sum_{i=1}^{B}\lambda_{GIOU}L_{GIOU}\left(\boldsymbol{b}_i,\boldsymbol{g}_i\right)+\lambda_{L_1}L_1\left(\boldsymbol{b}_i,\boldsymbol{g}_i\right), \tag{3.38}$$

where $\boldsymbol{b}_i$ is the vector composed of the top-left x-coordinate, the y-coordinate, and the width and height of the i-th predicted bounding box, and $\boldsymbol{g}_i$ is the vector comprising the top-left x-coordinate, the y-coordinate, and the width and height of the bounding box of the i-th training sample's ground-truth box. λ_{GIOU} and λ_{L_1} are non-negative hyperparameters, and B denotes the batch size of the training samples.

In some previous studies, researchers believed that joint learning of the localization and classification tasks may lead to suboptimal results. Thus, decoupling these two tasks is necessary. The training process is divided into two independent stages for the localization and classification tasks, which are optimized to achieve the best solution. In the first stage, the whole network, except for the candidate sample scoring head, is trained end-to-end using a localization-related loss function. This stage aims to ensure the object's inclusion in all search regions, thus improving the model's localization ability. The second stage focuses on optimizing the scoring head, and its loss function is defined as shown in the following equation:

$$L_{ce}=-\sum_{i=1}^{B}\left(l_i\,\mathbf{log}\left(s_i\right)+\left(1-l_i\right)\mathbf{log}\left(1-s_i\right)\right), \tag{3.39}$$

where l_i is the binary label of the i-th training sample, and s_i is the predicted probability of the i-th sample being the object. This two-stage training strategy enables the model to learn the key features of localization and classification separately, thus attaining high tracking precision and robustness. It enhances the model's object recognition and tracking accuracy in different settings and readies it for seamless integration into practical use.

4. Experiments

In this section, we introduce the comprehensive evaluation of the MCWTT tracking algorithm on three benchmark datasets: LaSOT [48], OTB100 [49], and UAV123 [50]. The hardware environment for the experiments was a server equipped with an Intel Core i9-12900K CPU and an Nvidia RTX 3090 24 GB GPU, with 128 GB of RAM. The software environment was based on Python 3.7 and PyTorch 1.8.2. During training, the model's basic training unit consisted of two templates and one search image. The input templates were 128 × 128 pixels, about twice the area of the target box, while the search region was 384 × 384 pixels, about five times the area of the target box. The backbone network was initialized with parameters from a pre-trained ResNet-50 network. The Transformer architecture comprised six layers of encoder-decoders with eight heads each, including MSA layers, MCA layers, and FFNs. Each layer of the encoder-decoders had eight heads, with 32 channels per head and a dropout rate of 0.1 to prevent overfitting. During model training, the loss function and generalized loss weights were optimized using the Adam [54] optimizer, with initial learning rates of 10^{-5} and 10^{-4} for the backbone network and other network components, respectively.

4.1. Quantitative analysis

Table 1. Detailed comparison of the LaSOT, OTB100, and UAV123 datasets.

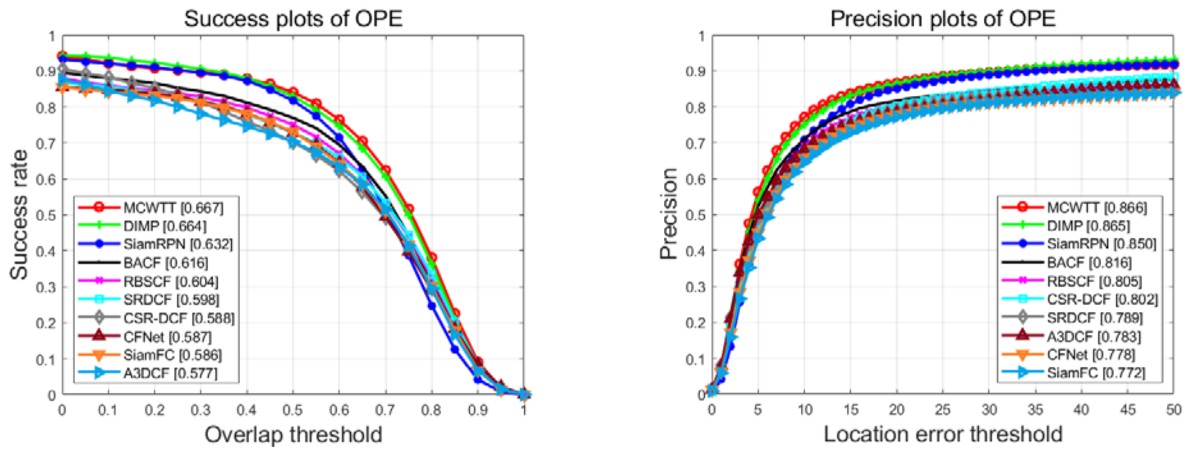
Method	**Year**	**LaSOT**			**OTB100**		**UAV123**	
		AUC(%)	P_{Norm}(%)	P(%)	SR(%)	PR (%)	AUC(%)	P(%)
SiamRPN++ [32]	2019	49.6	56.9	49.1	–	–	64.2	84.0
TransT [41]	2021	64.2	73.5	68.2	–	–	66.0	85.2
STARK [43]	2021	65.8	75.2	69.8	–	–	68.4	89.0
DSTrpn [56]	2021	43.4	54.4	–	64.6	85.7	–	–
CLNet*–BAN [57]	2022	52.9	62.3	52.6	–	–	–	–
TCTrack++ [58]	2022	43.5	48.4	41.4	54.3	72.0	51.9	73.1
RTSFormer [34]	2024	62.3	65.6	65.5	–	–	67.5	–
AGST–BR [59]	2024	56.7	–	58.3	–	–	66.3	–
SiamRPN++–ACM [60]	2024	52.3	–	–	71.2	–	–	–
MixFormer [55]	2024	69.6	79.9	75.9	71.6	94.4	68.7	89.5
MCWTT	Ours	65.6	74.5	70.0	66.7	86.6	68.7	89.2

4.1.1. LaSOT

LaSOT [48] is a large-scale tracking benchmark dataset with high-quality annotations and almost all real-world challenges, including fast motion, scale variation, and camera movement. It has 280 video sequences, with each frame labelled with high-quality bounding boxes. Tracker performance is evaluated using the area under the curve (AUC), the normalized precision (P_{Norm}), and precision (P) scores. AUC shows the success rate across IoU thresholds, P_{Norm} measures precision under the normalized distance, and P directly assesses the accuracy of the tracking results. As shown in Table 1, the MCWTT algorithm achieved an AUC score of 65.6%, a P_{Norm} score of 74.5%, and a P score of 70%, surpassing many advanced trackers. Though outperformed by STARK [43] and MixFormer [55], it remains competitive.

4.1.2. OTB100

The OTB100 [49] dataset is a widely utilized benchmark for evaluating object-tracking algorithms. Comprising 100 standardized video sequences with diverse challenging scenarios and detailed annotations, it assesses the robustness and adaptability of tracking algorithms. The evaluation follows the one-pass evaluation (OPE) protocol, tracking the entire sequence without restarts. This setup tests the tracker's performance in handling challenges like scale changes, occlusions, deformations, and motion blur. The tracking algorithms are ranked using success plots and precision plots, with success plots evaluated by the AUC, and precision plots assessed by the center position error (CPE).



(a) (b)

Figure 5. Tracker performance comparison on the OTB100 dataset. (a) Success plot; (b) precision plot.

Figure 5 shows a comparison of the proposed module with nine other trackers on the OTB100 dataset, including BACF [23], A3DCF [61], SRDCF [62], CSR-DCF [19], RBSCF [6], CFNet [63], DiMP [64], SiamFC [29], and SiamRPN [31]. As shown in Figure 5(a),(b), the MCWTT leads with a success rate of 66.7% and a precision of 86.6%, outperforming all the compared tracking methods while maintaining real-time capability. This validates the significant performance advantage of the proposed module in balancing real-time and accuracy requirements.

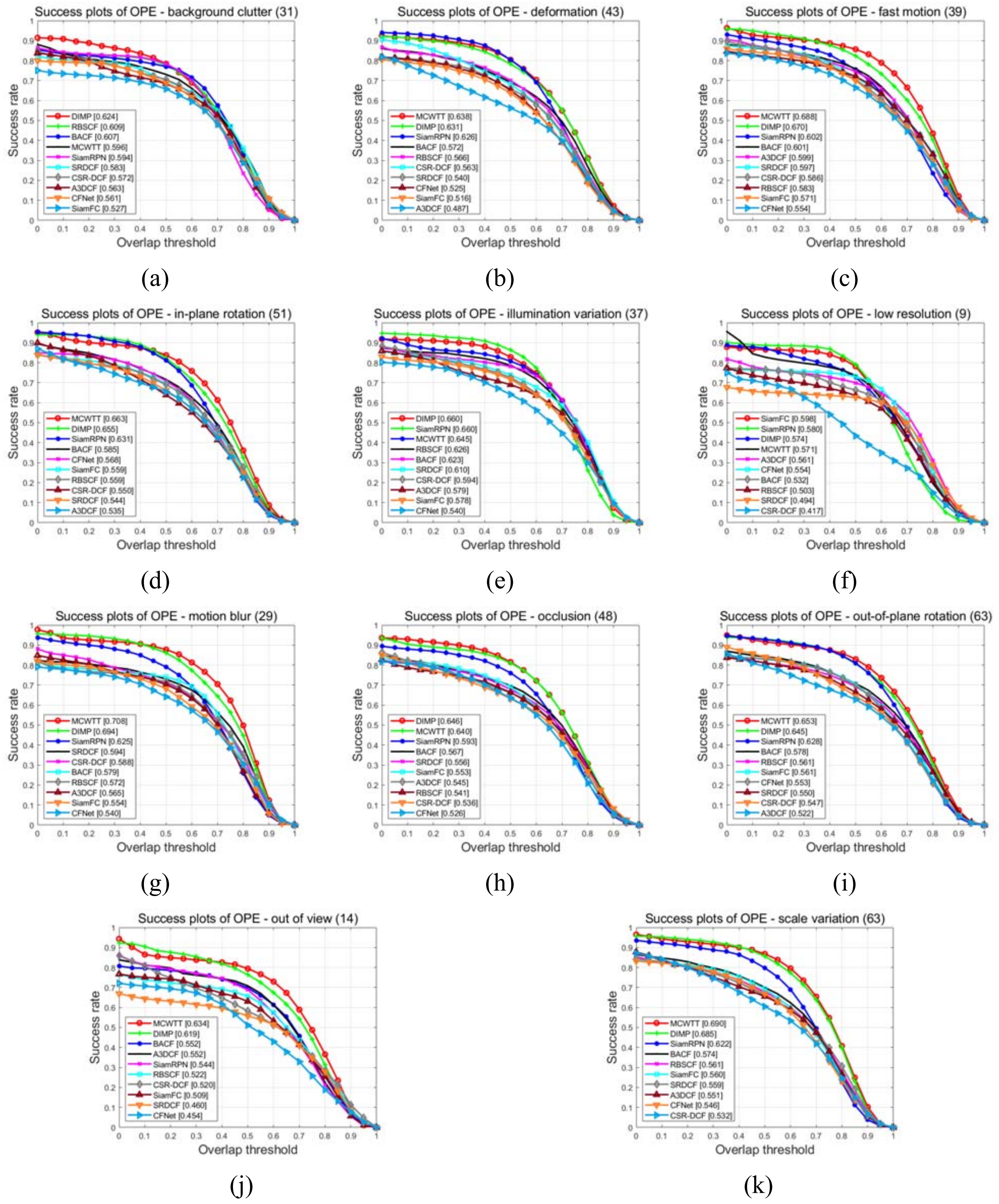


Figure 6. The success plots for various challenging attributes on the OTB100 dataset: (a) BC; (b) DEF; (c) FM; (d) IPR; (e) IV; (f) LR; (g) MB; (h) OCC; (i) OPR; (j) OV; (k) SV.

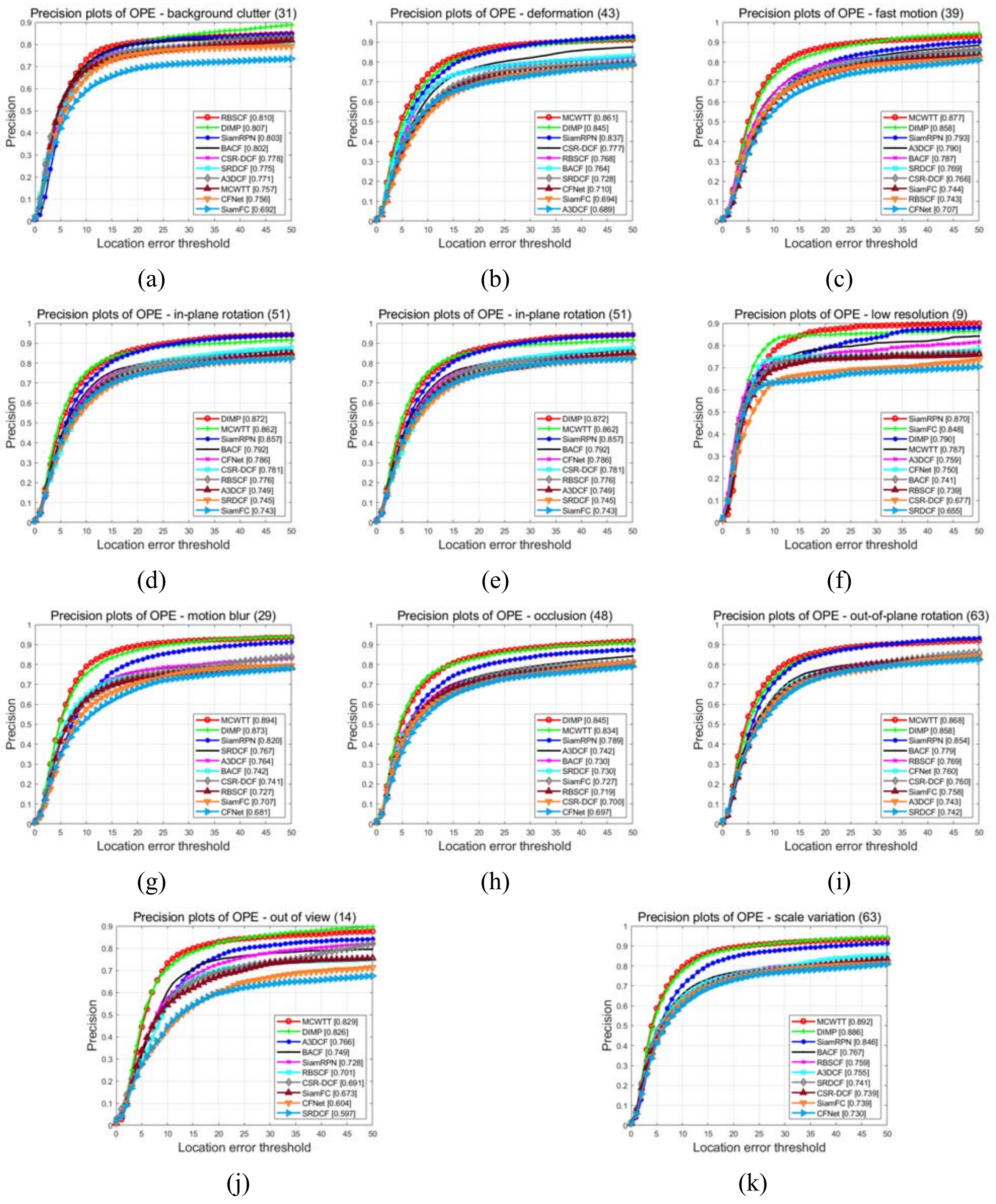


Figure 7. The precision plots for various challenging attributes on the OTB100 dataset: (a) BC; (b) DEF; (c) FM; (d) IPR; (e) IV; (f) LR; (g) MB; (h) OCC; (i) OPR; (j) OV; (k) SV.

Figures 6 and 7 compare the proposed module with nine other trackers across 11 visual challenge attributes of the OTB100 dataset, namely background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination variation (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out of view (OV), and scale variation (SV). Figure 6 shows the proposed module performs exceptionally well in multiple attributes, achieving success rates of 68.8% in the FM scenario and 63.4% in the OV scenario. Figure 7 shows the precision performance of the proposed module and other tracking algorithms under 11 different visual attribute challenges. The proposed module also performs exceptionally well in various attribute challenges, with 89.4% in the MB scenario and 89.2% in the SV scenario, significantly outperforming other competing algorithms and fully demonstrating the advantage of the proposed module in handling high-speed dynamic objects.

4.1.3. UAV123

The UAV123 [50] dataset contains 123 low-altitude unmanned aerial vehicle (UAV) video sequences, totaling over 110,000 frames, encompassing a variety of background environments ranging from urban landscapes to natural scenery. It is primarily used to evaluate the performance of different tracking algorithms. The dataset is divided into three subsets, each corresponding to distinct testing scenarios. The high-quality aerial video subset has 103 sequences shot by professional UAVs at heights of 5–25 meters, with frame rates of 30 to 96 frames per second (fps) and resolutions of 720P to 4K, ideal for tracking fast-moving targets. The low-cost UAV sub-set contains 12 sequences captured by economical UAVs with lower video quality, resolution, and more noise, increasing tracking difficulty. The synthetic data subset includes eight sequences generated by a UAV simulator to mimic real-world environmental changes. UAV123 can test a tracker's ability to handle fast motion, illumination changes, and occlusion, aiding in the development of systems stable in various environments. Evaluation metrics are the AUC of success plots and the CPE of precision plots.

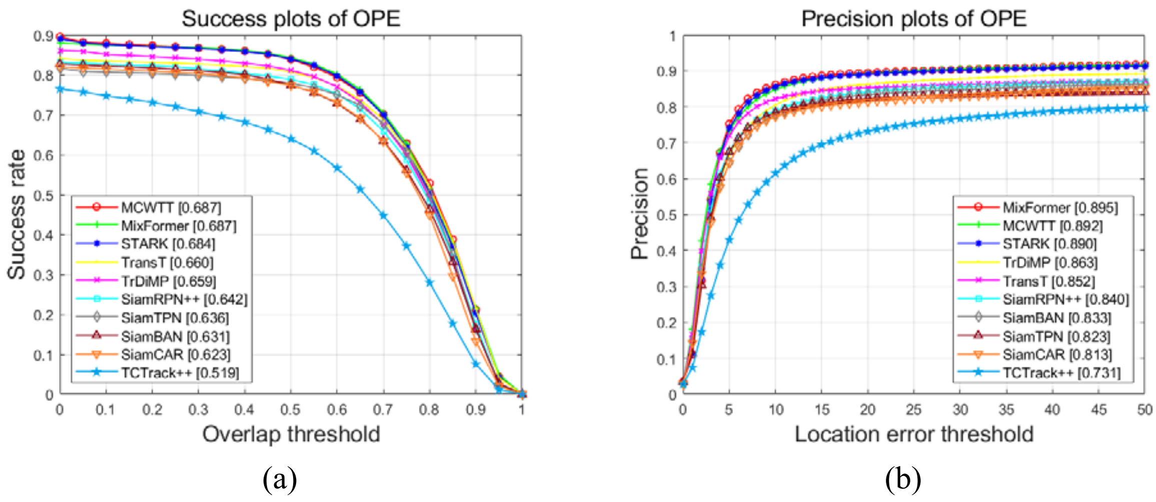


Figure 8. Tracker performance comparison on the UAV123 dataset. (a) Success plot; (b) precision plot.

Figure 8 compares the overall success and precision of the proposed module with the top tracking algorithms on the UAV123 dataset, including MixFormer [55], STARK [43], TransT [41], TCTrack++ [58], TrDiMP [65], SiamBAN [66], SiamCAR [67], SiamTPN [68], and SiamRPN++ [32]. In the success rate evaluation, the MCWTT and MixFormer both achieved 68.7%, ranking first. This indicates that the MCWTT's accuracy in object tracking is on par with the state-of-the-art methods. In the precision evaluation, the MCWTT's 89.2% is slightly lower than MixFormer's by 0.3% but surpasses other methods. This shows that MCWTT effectively balances real-time and accuracy requirements, demonstrating strong adaptability in complex scenes with occlusions, deformations, and illumination changes.

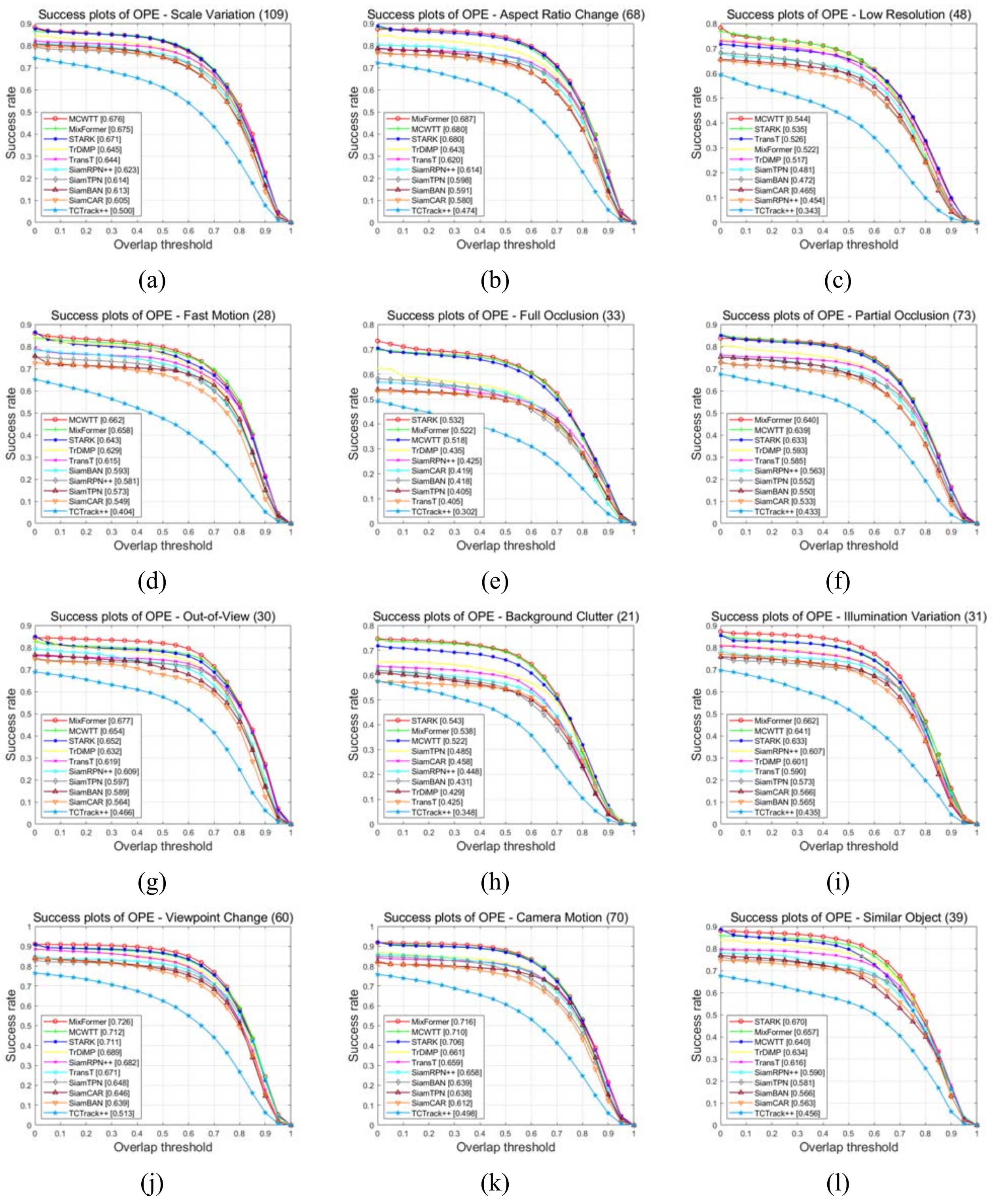


Figure 9. The success plots for various challenging attributes on the UAV123 dataset: (a) SV; (b) ARC; (c) LR; (d) FM; (e) FO; (f) PO; (g) OV; (h) BC; (i) IV; (j) VC; (k) CM; (l) SO.

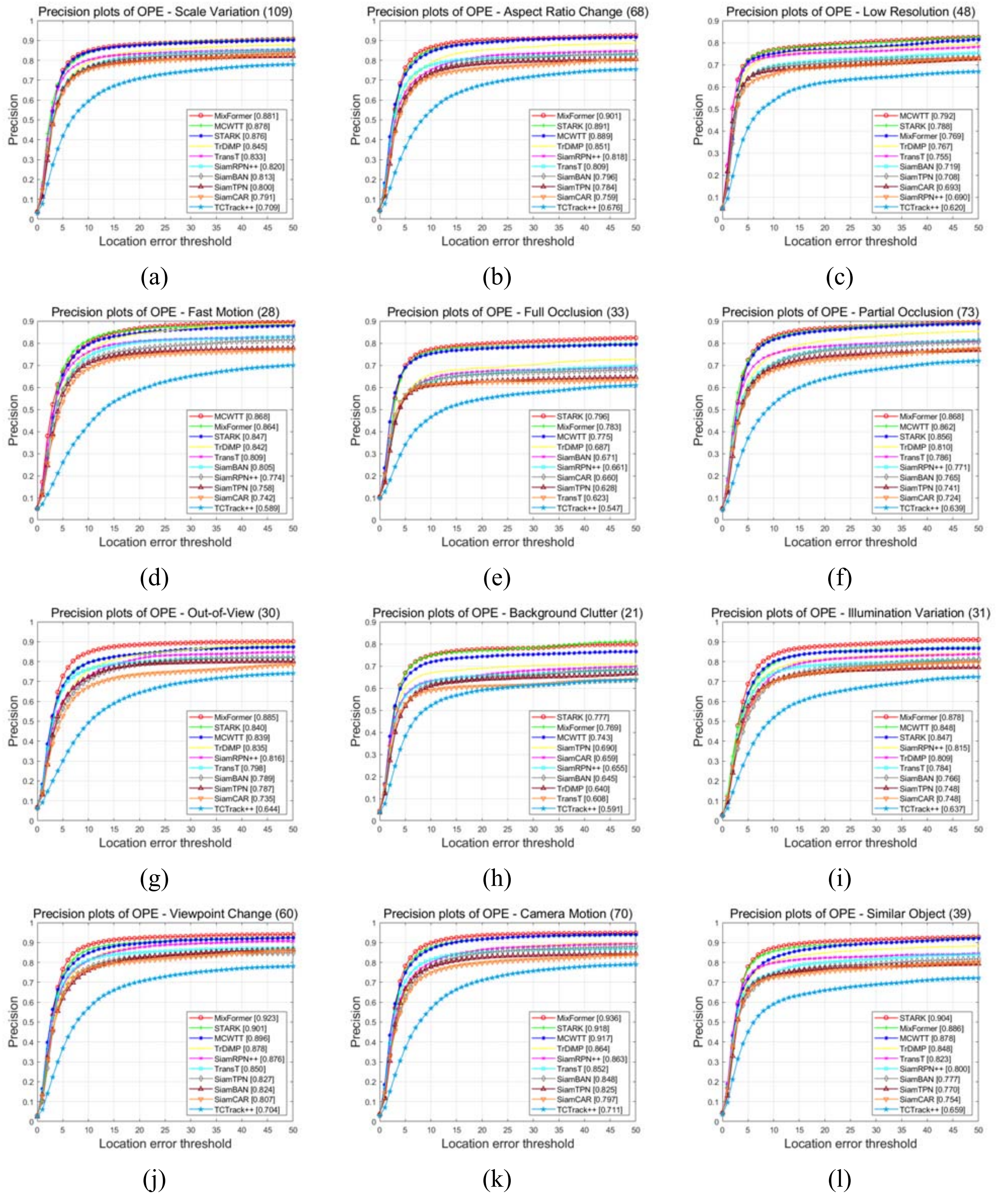


Figure 10. The precision plots for various challenging attributes on the UAV123 dataset: (a) SV; (b) ARC; (c) LR; (d) FM; (e) FO; (f) PO; (g) OV; (h) BC; (i) IV; (j) VC; (k) CM; (l) SO.

To demonstrate the tracking performance of the proposed module across diverse scenarios, we select 12 representative scene attributes from the UAV123 dataset for detailed analysis. These attributes encompass a variety of complex challenges, including scale variation (SV), aspect ratio change (ARC), low resolution (LR), fast motion (FM), full occlusion (FO), partial occlusion (PO), out-of-view (OV), background clutter (BC), illumination variation (IV), viewpoint change (VC), camera motion (CM), and similar object (SO).

As shown in Figures 9 and 10, the MCWTT exhibits robust tracking performance across all attribute scenarios. For instance, for the LR attribute (Figures 9 (a) and 10 (a)), the MCWTT achieves success and precision rates of 54.4% and 79.2%, respectively, outperforming all comparative algorithms. This highlights its exceptional adaptability to low-resolution objects. In FM scenarios, the MCWTT achieve success and precision rates of 66.2% and 86.8%, surpassing Mixformer by approximately 0.4% in both metrics. This demonstrates its superior efficiency and accuracy in tracking fast-moving objects. For highly challenging such as FO scenarios, the MCWTT maintains high tracking accuracy, with success and precision rates exceeding most benchmark algorithms. It ranks second only to Mixformer and STARK, further validating its reliability in handling full occlusions. These results underscore the MCWTT's versatility and robustness in addressing diverse real-world tracking challenges, particularly in scenarios involving rapid motion, low resolution, and occlusions.

4.2. Qualitative analysis

To better assess and demonstrate the performance of the MCWTT, we conducted frame-by-frame comparison tests with nine other trackers on the OTB100 dataset to show the tracking effects of each tracker in different scenes. We selected five videos representing typical tracking challenges, like fast motion, occlusion, deformation and scale variation, and illumination variation. The selected videos are "Diving_1", "Girl2_1", "Jump_1", "Skating2-1_1", and "Trans_1".

(1) **Fast motion and scale variation.** In the low-resolution scene of the "Diving_1" video sequence (Figure 11 (a)), the athlete briefly exhibits a mid-air flipping posture. In subsequent frames, the athlete plunges into the water with significant background changes. Many trackers were unable to effectively identify the rapidly moving object, while the module proposed in this paper accurately tracked the object throughout the entire video sequence. Similarly, in the "Jump_1" video sequence (Figure 11 (c)), the object undergoes scale variations during rapid motion. Only the proposed module successfully identified and accurately tracked the target across all frames of the full video sequence.

(2) **Occlusion.** In the "Girl2_1" video sequence shown in Figure 11(b), when the tracked girl is completely obscured by another pedestrian in Frame 107, only the proposed module accurately distinguishes the interfering object and continues to identify the object throughout subsequent occlusion, while other tracking methods either mistrack the interference object or experience tracking drift. This scenario fully demonstrates the superiority of the proposed module. Similarly, in the "Skating2-1_1" video sequence shown in Figure 11(d), the object athlete faces consecutive obstructions by the interfering athlete. The proposed module achieves stable tracking of the object throughout the video sequence. It adaptively resizes the bounding box to ensure the structural integrity of the object within the frame.

(3) **Deformation.** In the "Jump_1" video sequence shown in Figure 11(c), the athlete undergoes deformation during rapid motion, posing a challenge to the tracking algorithm. It is evident from Frame 95 of the video sequence that other tracking algorithms cannot track the gymnast's human form after landing. Only the proposed module can stably track the target throughout the entire video sequence, reflecting its effective handling of deformation and scale variation. Similarly, in the "Trans_1" video sequence shown in Figure 11(e), the proposed module delivers a more accurate identification of the

appearance contour of the object compared with other algorithms, significantly improving the accuracy of the tracking process.

(4) **Illumination Variation.** In the "Trans_1" video sequence shown in Figure 11(e), the object's background changes from bright to dark, making the object's contour more blurred than in previous scenes, such low-light conditions compromised the accuracy of several tracking algorithms, while the proposed method remained unaffected by illumination changes, accurately identifying the object.

In the above analysis, the proposed module shows better robustness and accuracy than the other trackers in complex tracking scenarios in low-resolution scenes, showing better robustness and accuracy.

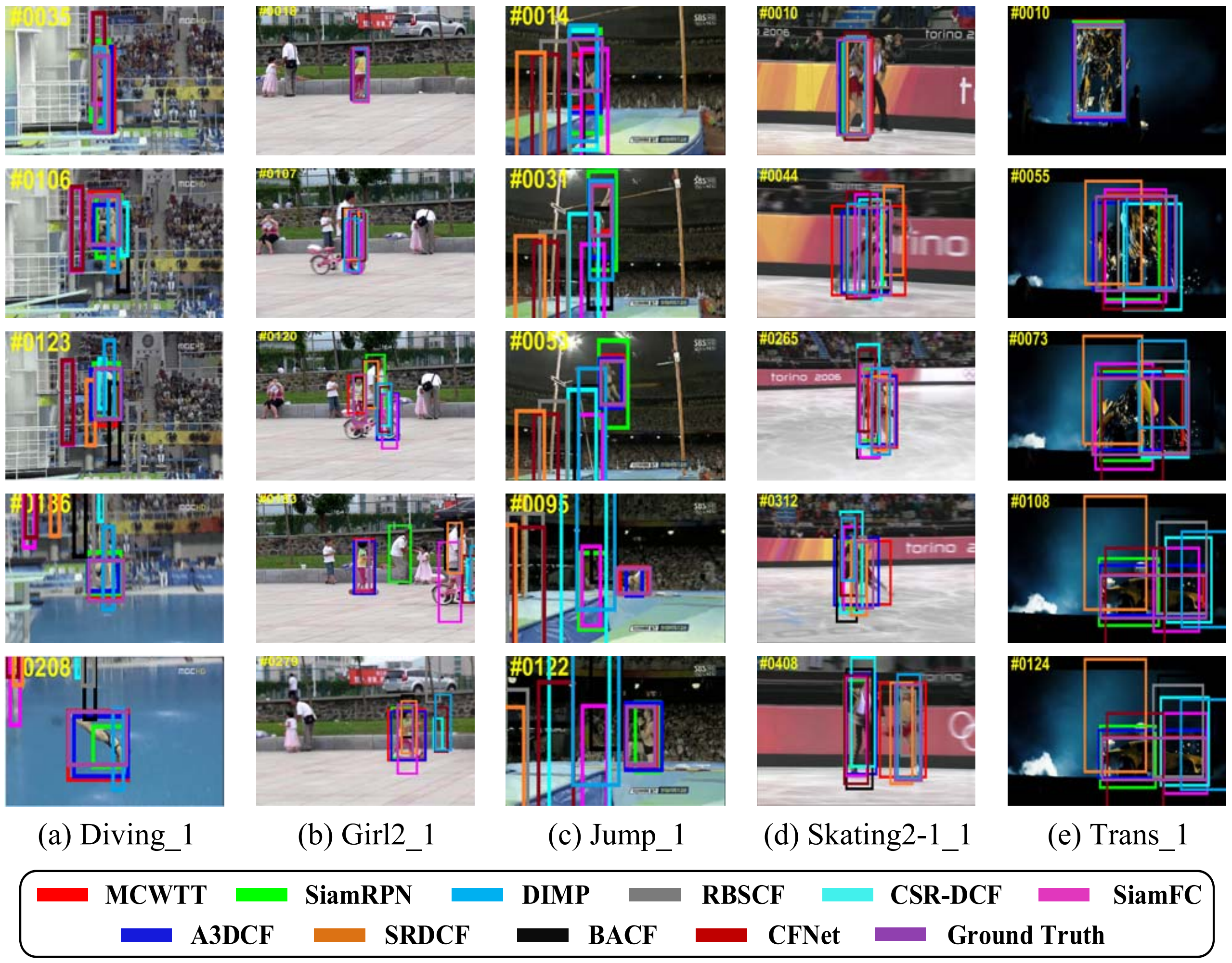


Figure 11. Visualization of tracking performance on different video sequences.

4.3. Ablation study

To assess the proposed module's specific enhancement of the tracking performance, we conduct a series of ablation experiments on the OTB100 dataset. By removing specific components from the structure while keeping others unchanged, the performance of the components is evaluated.

4.3.1. Ablation study on the FFM

In this subsection, we compare the model without the FFM (denoted as the CWTT) with the MCWTT to verify the FFM's effectiveness. As shown in Table 2, the MCWTT nearly matches the speed of the original model with only the SFM while reducing video random access memory (VRAM) usage, highlighting the advantage of the FFM in alleviating memory cache demands. The MCWTT effectively compresses the video memory through optimization in the model's architecture and computational efficiency, significantly reducing resource requirements without sacrificing tracking performance, thereby improving the performance on the OTB100 benchmark dataset. At the same time, the reduction in video memory occupancy makes the MCWTT more suitable for devices with limited video memory, further enhancing its advantages in practical applications.

Table 2. Comparison of VRAM usage and speed between the CWTT and MCWTT on the OTB100 dataset.

Method	VRAM usage (MiB)	Speed (fps)
CWTT	2318MiB	25.661
MCWTT	2172MiB	25.653

4.3.2. Ablation study on multi-scale windows

In this subsection, we replace the multi-scale windows with single-scale windows and compare the model with single-scale windows (denoted as the MCWTT-S) with the MCWTT to verify the effectiveness of multi-scale windows. As shown in Figure 12, the success rate and accuracy of the module with multi-scale window configuration are significantly improved compared with MCWTT-S.

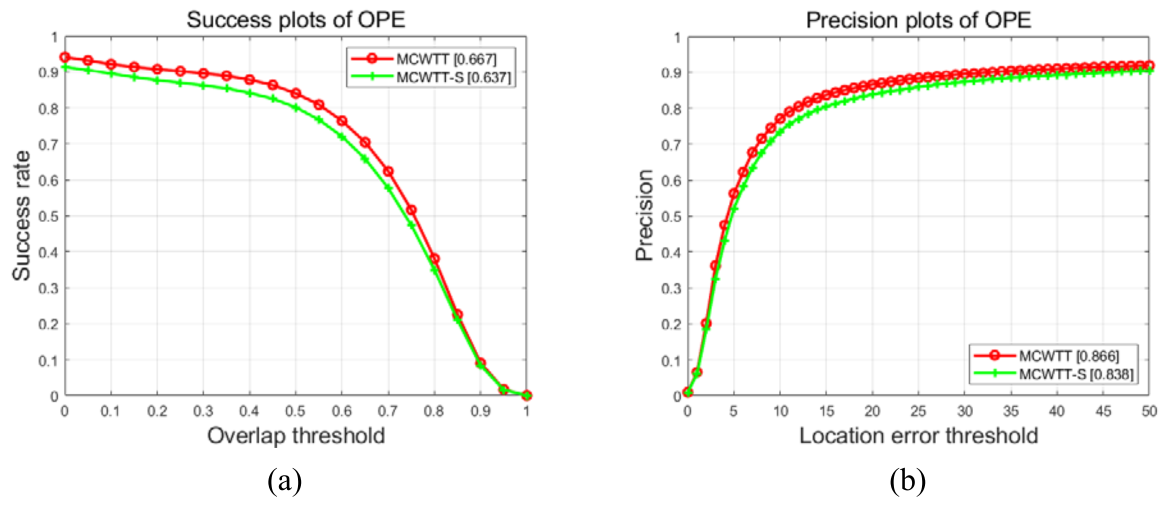


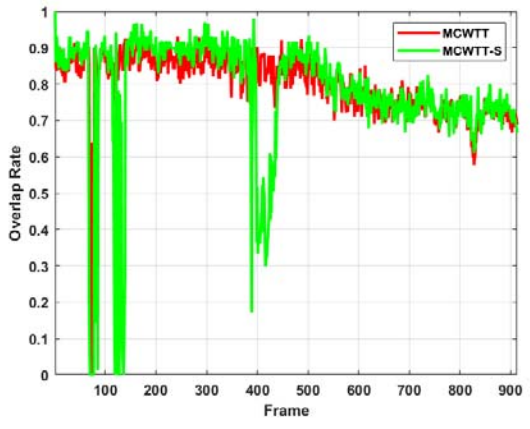
Figure 12. Performance comparison of the MCWTT and MCWTT-S models on the OTB100 dataset. (a) Success plot; (b) precision plot.

Table 3. Comparison of the MCWTT and MCWTT-S models' success rates on OTB100 dataset across 11 different challenge attributes.

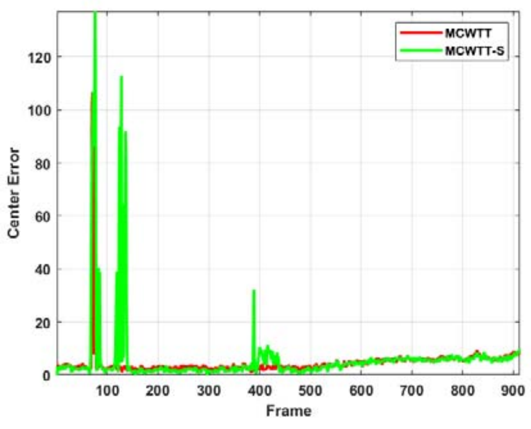
Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
MCWTT	0.596	0.638	0.688	0.663	0.645	0.571	0.708	0.640	0.653	0.634	0.690
MCWTT-S	0.541	0.646	0.653	0.613	0.614	0.575	0.674	0.592	0.614	0.550	0.647

Table 4. Comparison of the MCWTT and MCWTT-S models' precision rates on OTB100 dataset across 11 different challenge attributes.

Method	BC	DEF	FM	IPR	IV	LR	MB	OCC	OPR	OV	SV
MCWTT	0.757	0.861	0.877	0.862	0.818	0.787	0.894	0.834	0.868	0.829	0.892
MCWTT-S	0.712	0.878	0.837	0.826	0.792	0.826	0.859	0.773	0.826	0.719	0.844



(a)



(b)

Figure 13. Validation of the average overlap rate and central error rate curves of models with and without multi-scale windows in the "Car2" video sequence. (a) Average overlap rate; (b) central error rate.

Tables 3 and 4 list the success rates and accuracies of single-scale and multi-scale window configurations under different challenging scenarios. Combining the two proves that multi-scale windows are more adaptable to complex tracking environments and capture the complete information of the object more efficiently, reflecting on the tracking results.

We also compared the average overlap rate and center error rate curves of the MCWTT and MCWTT-S models under the "Car2" video sequence. As shown in Figure 13, the average overlap rate of the MCWTT is higher than that of the MCWTT-S and more stable. In terms of the center error rate, the MCWTT is lower than the MCWTT-S. The test results are also reflected in the visual tracking effects of the "Car2" video sequence in Figure 14. When there are interfering objects during the car's movement, the MCWTT-S loses the object, causing tracking drift and failure. In contrast, the MCWTT remains stable in tracking, proving the effectiveness of the multi-scale window configuration used in the MCWTT.

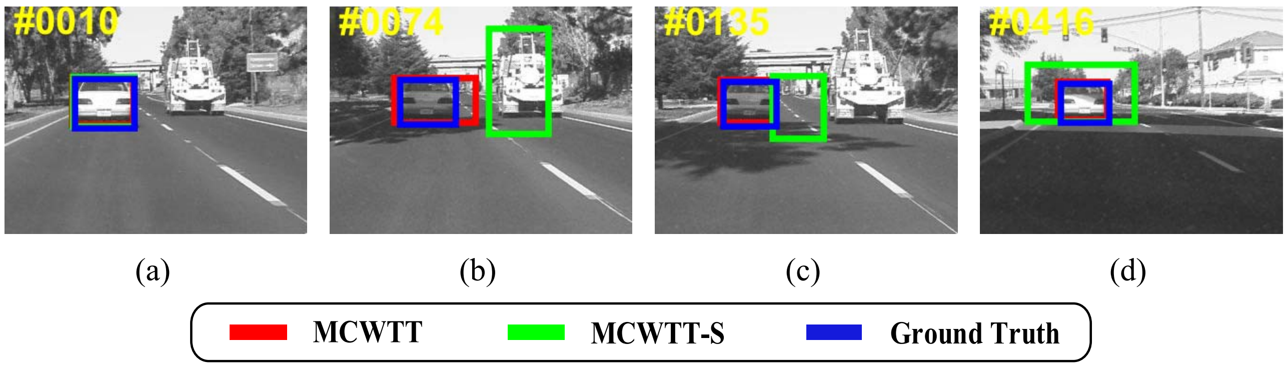


Figure 14. Comparison of models with and without multi-scale windows.

5. Conclusions

In this paper, we propose a multi-scale cyclic-shift window Transformer object tracker based on the fast Fourier transform (MCWTT). The proposed module introduces a cyclic-shift window mechanism to increase the diversity of sample positions. It also adopts a multi-scale window-level attention mechanism instead of the traditional single-scale pixel-level attention mechanism. The proposed module not only protects the integrity of the object but also enriches the diversity of training samples, further mining the object's location information and improving the tracking accuracy of the model. In addition, we use frequency-domain feature fusion instead of spatial-domain feature fusion, converting the attention matrix calculation between cyclic-shifted samples and non-cyclic-shifted samples into the frequency domain through the convolution theorem, effectively reducing the sample's storage and computational complexity and significantly improving the inference efficiency. Unlike the traditional encoder-decoder serial structure, we introduce a feature enhancement block in the network design, feeding it back to the feature fusion block to form a signal feedback loop, further enhancing the object state estimation ability and enabling the network to handle object tracking tasks in dynamic scenes more efficiently. Theoretical analysis and experimental verification show that the network has significant advantages in dealing with complex scenes, such as scale variation, background interference, and occlusion. However, there remains a performance gap compared with the state-of-the-art trackers. Our future work will focus on enhancing the tracker's capabilities through rigorous research. Ablation studies demonstrate that the cyclic shift operation effectively reduces the computational overhead for the module while maintaining its precision and efficiency. In real-world applications, enhancing interpretability and generalization while maintaining performance in harsh tracking conditions remains an open issue. Future research will focus on boosting the network's efficiency by streamlining the parameters without compromising tracking performance.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the Natural Science Foundation of Fujian Province (2024J01820, 2024J01821, 2024J01822), Natural Science Foundation Project of Zhangzhou City (ZZ2023J37), the Principal Foundation of Minnan Normal University (KJ19019), the High-level Science Research Project of Minnan Normal University (GJ19019), the Research Project on Education and Teaching of Undergraduate Colleges and Universities in Fujian Province (FBJY20230083), the Guangdong Province Natural Science Foundation (2024A1515011766) , and the State key laboratory major special projects of Jilin Province Science and Technology Development Plan (SKL202402024).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. Y. Li, X. Yuan, H. Wu, L. Zhang, R. Wang, J. Chen, CVT-track: Concentrating on valid tokens for one-stream tracking, *IEEE Trans. Circuits Syst. Video Technol.*, **34** (2024), 321–334. https://doi.org/10.1109/TCSVT.2024.3452231
2. S. Zhang, Y. Chen, ATM-DEN: Image inpainting via attention transfer module and decoder-encoder network, *SPIC*, **133** (2025), 117268. https://doi.org/10.1016/j.image.2025.117268
3. F. Chen, X. Wang, Y. Zhao, S. Lv, X. Niu, Visual object tracking: A survey, *Comput. Vision Image Underst.*, **222** (2022), 103508. https://doi.org/10.1016/j.cviu.2022.103508
4. F. Zhang, S. Ma, Z. Qiu, T. Qi, Learning target-aware background-suppressed correlation filters with dual regression for real-time UAV tracking, *Signal Process.*, **191** (2022), 108352. https://doi.org/10.1016/j.sigpro.2021.108352
5. S. Ma, B. Zhao, Z. Hou, W. Yu, L. Pu, X. Yang, SOCF: A correlation filter for real-time UAV tracking based on spatial disturbance suppression and object saliency-aware, *Expert Syst. Appl.*, **238** (2024), 122131. https://doi.org/10.1016/j.eswa.2023.122131
6. J. Lin, J. Peng, J. Chai, Real-time UAV correlation filter based on response-weighted background residual and spatio-temporal regularization, *IEEE Geosci. Remote Sens. Lett.*, **20** (2023), 1–5. https://doi.org/10.1109/LGRS.2023.3272522
7. J. Cao, H. Zhang, L. Jin, J. Lv, G. Hou, C. Zhang, A review of object tracking methods: From general field to autonomous vehicles, *Neurocomputing*, **585** (2024), 127635. https://doi.org/10.1016/j.neucom.2024.127635
8. X. Hao, Y. Xia, H. Yang, Z. Zuo, Asynchronous information fusion in intelligent driving systems for target tracking using cameras and radars, *IEEE Trans. Ind. Electron.*, **70** (2023), 2708–2717. https://doi.org/10.1109/TIE.2022.3169717
9. L. Liang, Z. Chen, L. Dai, S. Wang, Target signature network for small object tracking, *Eng. Appl. Artif. Intell.*, **138** (2024), 109445. https://doi.org/10.1016/j.engappai.2024.109445
10. R. Yao, L. Zhang, Y. Zhou, H. Zhu, J. Zhao, Z. Shao, Hyperspectral object tracking with dual-stream prompt, *IEEE Trans. Geosci. Remote Sens.*, **63** (2025), 1–12. https://doi.org/10.1109/TGRS.2024.3516833

11. N. K. Rathore, S. Pande, A. Purohit, An efficient visual tracking system based on extreme learning machine in the defence and military sector, *Def. Sci. J.*, **74** (2024), 643–650. https://doi.org/10.14429/dsj.74.19576
12. Y. Chen, Y. Tang, Y. Xiao, Q. Yuan, Y. Zhang, F. Liu, et al., Satellite video single object tracking: A systematic review and an oriented object tracking benchmark, *ISPRS J. Photogramm. Remote Sens.*, **210** (2024), 212–240. https://doi.org/10.1016/j.isprsjprs.2024.03.013
13. W. Cai, Q. Liu, Y. Wang, HIPTrack: Visual tracking with historical prompts, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2024), 19258–19267. https://doi.org/10.1109/CVPR52733.2024.01822
14. L. Sun, J. Zhang, D. Gao, B. Fan, Z. Fu, Occlusion-aware visual object tracking based on multi-template updating Siamese network, *Digit. Signal Process.*, **148** (2024), 104440. https://doi.org/10.1016/j.dsp.2024.104440
15. Y. Chen, L. Wang, eMoE-Tracker: Environmental MoE-based transformer for robust event-guided object tracking, *IEEE Robot. Autom. Lett.*, **10** (2025), 1393–1400. https://doi.org/10.1109/LRA.2024.3518305
16. Y. Sun, T. Wu, X. Peng, M. Li, D. Liu, Y. Liu, et al., Adaptive representation-aligned modeling for visual tracking, *Knowl. Based Syst.*, **309** (2025), 112847. https://doi.org/10.1016/j.knosys.2024.112847
17. J. Wang, S. Yang, Y. Wang, G. Yang, PPTtrack: Pyramid pooling based Transformer backbone for visual tracking, *Expert Syst. Appl.*, **249** (2024), 123716. https://doi.org/10.1016/j.eswa.2024.123716
18. C. Wu, J. Shen, K. Chen, Y. Chen, Y. Liao, UAV object tracking algorithm based on spatial saliency-aware correlation filter, *Electron. Res. Arch.*, **33** (2025), 1446–1475. https://doi.org/10.3934/era.2025068
19. A. Lukežič, T. Vojíř, L. Čehovin, J. Matas, M. Kristan, Discriminative correlation filter with channel and spatial reliability, *Int. J. Comput. Vision*, **126** (2018), 671–688. https://doi.org/10.1007/s11263-017-1061-3
20. T. Xu, Z. Feng, X. Wu, J. Kittler, Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking, *IEEE Trans. Image Process.*, **28** (2019), 5596–5609. https://doi.org/10.1109/TIP.2019.2919201
21. J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, *IEEE Trans. Pattern Anal. Mach. Intell.*, **37** (2015), 583–596. https://doi.org/10.1109/TPAMI.2014.2345390
22. E. O. Brigham, R. E. Morrow, The fast Fourier transform, *IEEE Spectrum*, **4** (1967), 63–70. https://doi.org/10.1109/MSPEC.1967.5217220
23. H. K. Galoogahi, A. Fagg, S. Lucey, Learning background-aware correlation filters for visual tracking, in *IEEE International Conference on Computer Vision (ICCV)*, (2017), 1144–1152. https://doi.org/10.1109/ICCV.2017.129
24. Z. Zhang, H. Peng, J. Fu, B. Li, W. Hu, Ocean: Object-aware anchor-free tracking, in *European Conference on Computer Vision (ECCV)*, (2020), 771–787. https://doi.org/10.1007/978-3-030-58589-1_46
25. Y. Zhang, H. Pan, J. Wang, Enabling deformation slack in tracking with temporally even correlation filters, *Neural Networks*, **181** (2025), 106839. https://doi.org/10.1016/j.neunet.2024.106839

26. Y. Chen, H. Wu, Z. Deng, J. Zhang, H. Wang, L. Wang, et al., Deep-feature-based asymmetrical background-aware correlation filter for object tracking, *Digit. Signal Process.*, **148** (2024), 104446. https://doi.org/10.1016/j.dsp.2024.104446
27. K. Chen, L. Wang, H. Wu, C. Wu, Y. Liao, Y. Chen, et al., Background-aware correlation filter for object tracking with deep CNN features, *Eng. Lett.*, **32** (2024), 1353–1363. https://doi.org/10.1016/j.dsp.2024.104446
28. J. Zhang, Y. He, W. Chen, L. D. Kuang, B. Zheng, CorrFormer: Context-aware tracking with cross-correlation and transformer, *Comput. Electr. Eng.*, **114** (2024), 109075. https://doi.org/10.1016/j.compeleceng.2024.109075
29. L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in *European Conference on Computer Vision (ECCV)*, (2016), 850–865. https://doi.org/10.1007/978-3-319-48881-3_56
30. Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, S. Wang, Learning dynamic siamese network for visual object tracking, in *IEEE International Conference on Computer Vision (ICCV)*, (2017), 1781–1789. https://doi.org/10.1109/ICCV.2017.196
31. B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2018), 8971–8980.
32. B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, J. Yan, SiamRPN++: Evolution of siamese visual tracking with very deep networks, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2019), 4277–4286.
33. L. Zhao, C. Fan, M. Li, Z. Zheng, X. Zhang, Global-local feature-mixed network with template update for visual tracking, *Pattern Recognit. Lett.*, **188** (2025), 111–116. https://doi.org/10.1016/j.patrec.2024.11.034
34. F. Gu, J. Lu, C. Cai, Q. Zhu, Z. Ju, RTSformer: A robust toroidal transformer with spatiotemporal features for visual tracking, *IEEE Trans. Human Mach. Syst.*, **54** (2024), 214–225. https://doi.org/10.1109/THMS.2024.3370582
35. O. Abdelaziz, M. Shehata, DMTrack: Learning deformable masked visual representations for single object tracking, *SIViP*, **19** (2025), 61. https://doi.org/10.1007/s11760-024-03713-0
36. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, in *the 31st International Conference on Neural Information Processing Systems (NIPS)*, (2017), 6000–6010.
37. O. C. Koyun, R. K. Keser, S. O. Şahin, D. Bulut, M. Yorulmaz, V. Yücesoy, et al., RamanFormer: A Transformer-based quantification approach for raman mixture components, *ACS Omega*, **9** (2024), 23241–23251. https://doi.org/10.1021/acsomega.3c09247
38. H. Fan, X. Wang, S. Li, H. Ling, Joint feature learning and relation modeling for tracking: A one-stream framework, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 341–357. https://doi.org/10.1007/978-3-031-20047-2_20
39. H. Zhang, J. Song, H. Liu, Y. Han, Y. Yang, H. Ma, AwareTrack: Object awareness for visual tracking via templates interaction, *Image Vision Comput.*, **154** (2025), 105363. https://doi.org/10.1016/j.imavis.2024.105363
40. Z. Wang, L. Yuan, Y. Ren, S. Zhang, H. Tian, ADSTrack: Adaptive dynamic sampling for visual tracking, *Complex Intell. Syst.*, **11** (2025), 79. https://doi.org/10.1007/s40747-024-01672-0

41. X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, H. Lu, Transformer tracking, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 8122–8131. https://doi.org/10.1109/CVPR46437.2021.00803
42. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., An image is worth 16x16 words: Transformers for image recognition at scale, preprint, arXiv:2010.11929. https://doi.org/10.48550/arXiv.2010.11929
43. B. Yan, H. Peng, J. Fu, D. Wang, H. Lu, Learning spatio-temporal transformer for visual tracking, in *IEEE International Conference on Computer Vision (ICCV)*, (2021), 10428–10437. https://doi.org/10.1109/ICCV48922.2021.01028
44. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, et al., Swin Transformer: Hierarchical vision transformer using shifted windows, in *IEEE/CVF International Conference on Computer Vision (ICCV)*, (2021), 10012–10022. https://doi.org/10.1109/ICCV48922.2021.00986
45. L. Lin, H. Fan, Z. Zhang, Y. Xu, H. Ling, SwinTrack: A simple and strong baseline for transformer tracking, in *Advances in Neural Information Processing Systems (NIPS)*, **35** (2022), 16743–16754.
46. Z. Song, J. Yu, Y. P. P. Chen, W. Yang, Transformer tracking with cyclic shifting window attention, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2022), 8781–8790. https://doi.org/10.1109/CVPR52688.2022.00859
47. Y. Chen, K. Chen, Four mathematical modeling forms for correlation filter object tracking algorithms and the fast calculation for the filter, *Electron. Res. Arch.*, **32** (2024), 4684–4714. https://doi.org/10.3934/era.2024213
48. H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, et al., LaSOT: A high-quality benchmark for large-scale single object tracking, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2019), 5369–5378. https://doi.org/10.1109/CVPR.2019.00552
49. Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2013), 2411–2418. https://doi.org/10.1109/CVPR.2013.312
50. M. Mueller, N. Smith, B. Ghanem, A benchmark and simulator for UAV tracking, in *Computer Vision–ECCV 2016*, (2016), 445–461. https://doi.org/10.1007/978-3-319-46448-0_27
51. Y. Huang, Y. Chen, C. Lin, Q. Hu, J. Song, Visual attention learning and antiocclusion-based correlation filter for visual object tracking, *J. Electron. Imaging*, **32** (2023), 23. https://doi.org/10.1117/1.JEI.32.1.013023
52. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016), 770–778.
53. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in *European Conference on Computer Vision (ECCV)*, (2020), 213–229. https://doi.org/10.1007/978-3-030-58452-8_13
54. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980
55. Y. Cui, C. Jiang, G. Wu, L. Wang, MixFormer: End-to-end tracking with iterative mixed attention, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 4129–4146. https://doi.org/10.1109/TPAMI.2024.3349519
56. J. Shen, Y. Liu, X. Dong, X. Lu, F.S. Khan, S. Hoi, Distilled siamese networks for visual tracking, *IEEE Trans. Pattern Anal. Mach. Intell.*, **44** (2022), 8896–8909. https://doi.org/10.1109/TPAMI.2021.3127492

57. X. Dong, J. Shen, F. Porikli, J. Luo, L. Shao, Adaptive siamese tracking with a compact latent network, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2023), 8049–8062. https://doi.org/10.1109/TPAMI.2022.3230064
58. Z. Cao, Z. Huang, L. Pan, S. Zhang, Z. Liu, C. Fu, Towards real-world visual tracking with temporal contexts, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2023), 15834–15849. https://doi.org/10.1109/TPAMI.2023.3307174
59. Y. Yang, X. Gu, Attention-based gating network for robust segmentation tracking, *IEEE Trans. Circuits Syst. Video Technol.*, **35** (2025), 245–258. https://doi.org/10.1109/TCSVT.2024.3460400
60. W. Han, X. Dong, Y. Zhang, D. Crandall, C. Z. Xu, J. Shen, Asymmetric Convolution: An efficient and generalized method to fuse feature maps in multiple vision tasks, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 7363–7376. https://doi.org/10.1109/TPAMI.2024.3400873
61. X. Zhu, Y. Wu, D. Xu, Z. Feng, J. Kittler, Robust visual object tracking via adaptive attribute-aware discriminative correlation filters, *IEEE Trans. Multimedia*, **23** (2021), 2625–2638. https://doi.org/10.1109/TMM.2021.3050073
62. M. Danelljan, H. Gustav, F. Shahbaz Khan, M. Felsberg, Learning spatially regularized correlation filters for visual tracking, in *IEEE International Conference on Computer Vision (ICCV)*, (2015), 4310–4318. https://doi.org/10.1109/ICCV.2015.490
63. J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P. H. S. Torr, End-to-end representation learning for correlation filter based tracking, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2017), 5000–5008. https://doi.org/10.1109/CVPR.2017.531
64. G. Bhat, M. Danelljan, L.V. Gool, R. Timofte, Learning discriminative model prediction for tracking, in *IEEE/CVF International Conference on Computer Vision (ICCV)*, (2019), 6182–6191. https://doi.org/10.1109/ICCV.2019.00628
65. N. Wang, W. Zhou, J. Wang, H. Li, Transformer meets tracker: exploiting temporal context for robust visual tracking, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2021), 1571–1580. https://doi.org/10.1109/CVPR46437.2021.00162
66. Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 6668–6677. https://doi.org/10.1109/CVPR42600.2020.00670
67. Y. Guo, H. Li, L. Zhang, L. Zhang, K. Deng, F. Porikli, SiamCAR: Siamese fully convolutional classification and regression for visual tracking, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (2020), 1176–1185. https://doi.org/10.1109/CVPR42600.2020.00630
68. D. Xing, N. Evangeliou, A. Tsoukalas, A. Tzes, Siamese transformer pyramid networks for real-time UAV tracking, in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, (2022), 1898–1907. https://doi.org/10.1109/WACV51458.2022.00196