A Coarse-to-Fine Object Tracking Based on Attention Mechanism

Haixi Wen¹, Xiaoming Chen*¹ and Li Zhou¹
¹ College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu Province, 211106, China
*Corresponding author’s e-mail: xmchen@nuaa.edu.cn

Abstract. In the visual object tracking task, although the convolutional neural network has excellent performance, it is difficult to learn the global and long-range semantic information interaction due to the inherent locality of convolution. Simultaneously, the existing template updating methods focus on the target position of the next frame, which will integrate more irrelevant background and cause tracking instability. Therefore, to address this issue, inspired by Transformer, in this paper, we propose a coarse-to-fine tracking method based on the attention mechanism (CTFT). First, employing swin transformer block instead of the original convolutional network backbone to achieve self-attention from local to global. Second, using attention to effectively combine template and search region features in the part of feature fusion, and explore the possibility of using attention mechanism as the backbone in object tracking for the first time. Finally, a tracking strategy from coarse to fine is proposed. In the offline coarse tracking stage, the initial estimation of the target object is generated to obtain the coarse regress bounding box of the target. In the online fine tracking stage, we make use of the coarse regress bounding box to expand the corresponding object frame, and use the segmentation to get the fine regress bounding box of the object. Experiments on challenging benchmarks including GOT-10k, LaSOT, VOT2018, VOT2019, UAV123 and OTB-100 demonstrate that the proposed CTFT outperforms many state-of-the-art trackers and achieves leading performance.

1. Introduction
Visual object tracking aims to track a given target object at each frame over a video sequence. It is a fundamental task in computer vision [1, 2, 3], and has numerous practical applications. However, developing a fast, accurate and robust tracker is still highly challenging due to the vast number of deformations, motions and occlusions that often occur on video objects with complex background [4].

Recently, the Siamese networks developed based on similarity comparison strategies have attracted great attention of the visual tracking community. These Siamese trackers learn a general similarity map by cross-correlation between the target template and the search area extracted through the backbone network, and formalize the visual tracking problem. However, due to the inherent locality of the convolution operation, it is difficult to learn the global and long-range semantic information interaction. Therefore, this naturally leads to an interesting question: Is there a better feature extraction method than convolution neural network? In the original Siamese tracker [5], the object template is initialized only in the first frame. During the target tracking process, the template remains fixed, and the rest of the video is matched with the unchanged initial frame. However, the appearance changes are often large when the target is in motion, and the unchanged template may cause the tracker tracking failure. To accommodate this problem, the recent Siamese trackers [6,7] implement a simple linear update strategy by using a
running average with a fixed learning rate. In addition, this update method is constant in all dimensions and cannot be updated locally, which is very harmful in the case of occlusion, resulting in a large amount of irrelevant background information into the template and cause tracking failure.

In this paper, in order to solve the above two problems, we designed a coarse-to-fine target tracking method based on the attention mechanism, and proposed a new tracking algorithm (named CTFT), which uses swin transform block instead of convolution neural network as the backbone to solve the inherent locality of convolution operation. The main contributions of this paper are as follows:

- We propose a new tracker that uses swin transform block to replace the backbone of the convolutional neural network to solve the inherent locality of the convolution operation to obtain better quality samples and search area feature maps, which provides a precondition for feature fusion.
- We propose two-stage tracking, which is divided into offline coarse tracking and online fine tracking. Through the processing of the coarse regress bounding box obtained from offline coarse tracking, the segmentation is used in the online fine tracking stage to obtain the fine tracking regress bounding box of the corresponding object to obtain higher accuracy and stability.
- We have conducted comprehensive experiments on large-scale benchmark datasets including GOT-10k, LaSOT, VOT2018, VOT2019, UAV123 and OTB-100. The experiments show that our tracker is superior to the most advanced algorithms, while achieving the running speed of about 35 FPS satisfies real-time performance.

2. Transformer Tracking

This section presents the two-stage tracking architecture for visual tracking, named CTFT. As shown in Figure 1, our CTFT is consist of four components: backbone network, feature fusion network, coarse regress bounding box prediction head and online fine tracking.

2.1. Attention Mechanism

2.1.1. Encoder.

The encoder has a standard architecture and consists of a multi-head self-attention module and a feed forward network (FFN). Efficiency is very important for visual tracking scenarios. As shown in the left of Figure 2, We find that deleting FFN will not have a very serious impact on performance. Therefore, in order to achieve a good balance between speed and performance and streamline the network structure, the encoder only contains a multi-head self-attention module and residual structure. We assume there is a template image feature map $Z_0 \in \mathbb{R}^{d \times H \times W}$, where $H$ and $W$ are the height and width of the feature map, and $d$ is channel dimension. Then, we collapse the spatial dimensions of $Z_0$ into one dimension.
as a sequence with the size of $\mathbb{R}^{d_x \times H_s \times W_s}$. Since the transformer architecture is permutation-invariant, we use a sine function to generate spatial positional encoding [10].

Figure 2 The left side of the figure is the encoding module, which deletes the FNN layer to ensure a good balance between speed and performance. The right side of the figure is a more detailed decoding module that performs feature fusion by accepting features from different branches. Spatial position coding is used to encode position information.

2.1.2. Decoder.
The decoder also follows the standard architecture, which contains multi-headed attention modules for self- and encoder-decoder (cross-) attention followed by a FFN. As shown in the right of Figure 2. Different from the original decoding framework, inspired by [12], we adopt double cross attention. Two cross-feature augment modules receive the feature maps of their own and the other branch at the same time and fuse these two feature maps through multi-head cross-attention. The input of decoder is a search feature map $X_0 \in \mathbb{R}^{d_x \times H_s \times W_s}$, of which the channel dimension is same with $z_0$, but the spatial dimensions are larger ($W_1 > W_s$, $H_1 > H_s$) for tracking purpose. Similar to the encoder, the spatial dimensions of $X_0$ is also collapsed into one dimension, resulting in a sequence of $\mathbb{R}^{d_x \times H_s \times W_s}$. The fixed positional encodings are also added in this sequence.

2.1.3. Multi-head Attention.
Attention is the fundamental component in designing our feature fusion network. Given queries Q, keys K and values V, the attention function is the scale dot-product attention, defined in equation (1).

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

Where $d_k$ is the key dimensionality.

In order to enable the model to focus on different aspects of information and enable the mechanism to consider various attention distributions, [9] extends the attention mechanism (1) to multiple heads, and defines the mechanism of multi-head attention as shown in equation (2), readers can check the reference [9] for a more detailed description.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, ..., H_m)V^O$$

$$H_i = \text{Attention}(QW_i^O, KW_i^K, VW_i^V)$$

Where $W_i^O \in \mathbb{R}^{d_{x\times d_k} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{x\times d_k} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{x\times d_k} \times d_k}$, and $W_i^O \in \mathbb{R}^{d_{x\times d_k} \times d_k}$ are parameter matrices. In this work, we employ $n_h = 8$, $d_m = 256$ and $d_k = d_v = d_m / n_h = 32$ as default values.
As shown in the encoder on the left of Figure 2, the encoder collects and integrates information from different positions of the feature map by using multi-head self-attention and residual structure. However, as shown in equation (1), the attention mechanism cannot distinguish the position information of the input feature sequence. Thus, we introduce a spatial positional encoding process to the input $X \in \mathbb{R}^{d \times N_i}$. We use a sine function to generate spatial positional encoding [10]. Finally, the mechanism of encoder can be summarized as

$$X_{eo} = X + \text{MultiHead}(X + P_{eo}, X + P_{eo}, X)$$

(4)

Where $P_{eo} \in \mathbb{R}^{d \times N_i}$ is the spatial positional encodings and $X_{eo} \in \mathbb{R}^{d \times N_i}$ is the out put of encoder.

The structure of decoder is shown in the right of Figure 2. decoder fuses the feature vectors from two inputs by using multi-head cross-attention in the form of residual. Similar to encoder, spatial positional encoding is also used in decoder. In addition, a FFN module is used to enhance the fitting ability of the model, which is a fully connected feed-forward network that consists of two linear transformation with a ReLU in between, the equation is as follows (5)

$$\text{FFN}(x) = \max(0, xW + b)W + b$$

(5)

The symbols $W$ and $b$ stand for weight matrices and basis vectors, respectively. The subscripts denote different layers. Thus, the mechanism of decoder can be summarized as

$$\hat{X}_{do} = X_q + \text{MultiHead}(X_q + P_{qo}, X_{kv} + P_{vo}, X_{kv})$$

(6)

$$X_{do} = \hat{X}_{do} + \text{FFN}(\hat{X}_{do})$$

(7)

Where $X_q \in \mathbb{R}^{d \times N_q}$is the input of the branch where the module is applied, $P_q \in \mathbb{R}^{d \times N_q}$ is the spatial positional encoding corresponding to $X_q$. $X_{kv} \in \mathbb{R}^{d \times N_{kv}}$ is the input from another branch, and $P_{kv} \in \mathbb{R}^{d \times N_{kv}}$ is the spatial encoding for the coordinate of $X_{kv}$. $X_{do} \in \mathbb{R}^{d \times N_{do}}$ is the output of decoder.

### 2.2. Swin Transformer block

The traditional standard transformer encoder [9] is stacked by the same modules, as shown in (a) of the right part of Figure 3, each module is composed of two LayerNorm (LN) layers, a Multi-head Self Attention (MSA) and a Multi Layer Perceptron (MLP), while adding a residual structure to each MSA and MLP module, the traditional standard transformer encoder formulas are as follows

$$\hat{z}_t = \text{MSA}(\text{LN}(z_{t-1})) + z_{t-1}$$

(8)

$$z_t = \text{MLP}(\text{LN}(\hat{z}_t)) + \hat{z}_t$$

(9)

Figure 3 Feature extraction backbone (left) is composed of swin transformer block, we only use the third stage (discussed in section 2.3). On the right side of the figure (a) is the traditional standard transformer encoder [9], and on the right side of the figure (b) is the architecture of swin transformer module.

In the standard Transformer architecture, the computational complexity is the second power of the number of tokens, which makes the task of high-resolution image less efficient and is fatal for target tracking. In order to have a more efficient model, swin transformer [11] proposes Window based MSA (W-MSA) and Shifted Window based MSA (SW-MSA). As shown in the left side of Figure 3, we use the swin transformer block to replace the convolutional neural network as the backbone, this backbone is composed of three stages, and in (b) of the right part of Figure 3, each swin transformer block is
composed of LayerNorm (LN) layer, multi-head self attention module, residual connection and 2-layer MLP with GELU non-linearity. W-MSA divides the input features into non-overlapping windows to conduct self-attention on the local windows. \( \hat{z}_l \) and \( z_l \) represent the output of W-MSA and MLP in the \( l \)-th layer, respectively. The formulas are as follows

\[
\hat{z}_l = W - \text{MSA}(\text{LN}(z_{l,1})) + z_{l,1}
\]

\[
z_l = \text{MLP}(\text{LN}(\hat{z}_l)) + \hat{z}_l
\]

The disadvantage of W-MSA is that there is only the self-attention of the local window, and the lack of information interaction between each window. In order not to add additional calculations, SW-MSA is introduced after W-MSA to solve this problem.

The window configuration of SW-MSA is completely different from the previous W-MSA layer. It proposes an efficient batch processing method by cyclic-shifting to the upper-left. After this shift, several sub-windows are not adjacent in the feature map form a batch window, and then cyclically shift to ensure efficiency, the outputs of SW-MSA and MLP module can be written as

\[
\hat{z}_{l,1} = \text{SW-MSA}(\text{LN}(z_{l,1}))) + z_{l,1}
\]

\[
z_{l,1} = \text{MLP}(\text{LN}(\hat{z}_{l,1})) + \hat{z}_{l,1}
\]

### 2.3. Overall Architecture

In this section, we will introduce the specific structure and parameters of each structure in detail.

#### 2.3.1. Feature Extraction.

The tracker based on the swin transformer we proposed takes a picture pair input like Siamese-based tracker [8] into the backbone network, which are template image patch \( z \in \mathbb{R}^{3 \times H_0 \times W_0} \) and search region image patch \( x \in \mathbb{R}^{3 \times H_0 \times W_0} \). The bounding box of the template is doubled in the center of the first frame of the tracking video to obtain information about the appearance of the target and the surrounding background. The search region patch in order to obtain as much as possible the range that the target may move, we use the target center as the origin to expand the target regress bounding box obtained in the previous frame to four times and at the same time reshape the template and the search region patch to square, respectively. The backbone processes the search region and the template to obtain their features maps \( f_z \in \mathbb{R}^{C \times H_z \times W_z} \) and \( f_x \in \mathbb{R}^{C \times H_x \times W_x} \). \( H_z, W_z = \frac{H_0}{16}, \frac{W_0}{16} \), \( H_x, W_x = \frac{H_0}{16}, \frac{W_0}{16} \) and \( C = 1024 \).

#### 2.3.2. Feature Fusion.

The encoder and decoder (described in 2.1) are used to effectively fuse the extracted features \( f_z \) and \( f_x \). The extracted features are first subjected to a \( 1 \times 1 \) convolution to reduce the number of channels. The \( f_z \) and \( f_x \) dimensions are reduced to \( f_{z,0} \in \mathbb{R}^{d \times H_z \times W_z} \) and \( f_{x,0} \in \mathbb{R}^{d \times H_x \times W_x} \), where \( d = 256 \). It can be seen from 3.1 that the input of the encoder and the decoder is a set of feature vectors, so we also need to flatten \( f_{z,0} \) and \( f_{x,0} \) in the spatial dimension, so we can get \( f_{z,1} \in \mathbb{R}^{d \times H_z W_z} \) and \( f_{x,1} \in \mathbb{R}^{d \times H_x W_x} \). We input the processed features \( f_{z,1} \) and \( f_{x,1} \) into the encoder and decoder. The encoder performs self-context data enhancement to focus on the main content of the picture. Two cross-focus fusion devices perform feature fusion to find the most matching feature, and finally decode and get a feature map \( f \in \mathbb{R}^{d \times H_z W_z} \).

#### 2.3.3. Prediction Head.

The predictor is composed of one of the simplest and most effective three-layer perceptrons. The perceptron includes hidden dimension \( d \) and activation function ReLU. It contains two branches, namely classification branch and coarse regression branch. The feature graph \( f \in \mathbb{R}^{d \times H_z W_z} \) obtained by the backbone gets the result \((H, W)\) through the three-layer perceptron. The classification branch...
judges whether the foreground and background are our tracking targets. The rough regression branch completely discards the anchor point or anchor box based on prior knowledge, which can directly predict the normalized coordinates and select the regression coordinates corresponding to the point with the highest score of the classification branch and make the tracker more concise.

2.3.4. online fine tracking.
Inspired by deepmask [13], we designed an online fine tracker. As shown in figure 4, when the prediction head gets the coarse regress bounding box, the enlarged image block is obtained by expanding the coarse regress bounding box of the current frame search area by 1.7 times. The trained deepmask performs target segmentation on the enlarged image patch to obtain the mask binary image of the image patch, and then we use morphological operations to select the mask with the largest connected domain, and use an ellipse approach to approximate the pose of the target object and the center of the ellipse. The point is the intersection of the diagonals of the target box, and its long and short axes are used as the length and width of the bounding box. The resulting rotated rectangular box is usually slightly larger than the true value, and only need to zoom it slightly to get a fine regress bounding box.

Figure 4 Online fine tracking structure. The coarse regression box is re-segmented to get the fine regression box to improve the accuracy.

2.4. Training Loss
Through the binary classification and regression results obtained by the prediction head, we select the prediction corresponding to the feature vector within the ground truth box as a positive sample, and the prediction outside the range is a negative sample. All samples will produce classification loss, while only positive samples will produce regression loss. In order to ensure the balance between samples, the negative sample loss will be reduced by 16 times. We use the standard binary cross-entropy loss for classification, which is defined as

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

(14)

Where \(y_j\) denotes the ground-truth label of the \(j\)-th sample, \(y_j = 1\) denotes foreground, and \(p_j\) denotes the probability belong to the foreground predicted by the learned model.

For regression, we follow [14] apply a linear combination of \(\ell_1-\text{norm}\) loss \(L_i(.,.)\) and the generalized IoU loss \(L_{\text{GIOU}}(.,.)\). The regression loss can be formulated as

\[
L_{\text{reg}} = \sum_j 1_{y_j=1} [\lambda_g L_{\text{GIOU}}(b_j, \hat{b}) + \lambda_i L_i(b_j, \hat{b})]
\]

(15)

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

3.1. Implementation Details
We train our model on the LaSOT and GOT10k datasets. The backbone network swin transformer block is initialized with the pre-processed parameters on swin transformer [11], and the other parameters are initialized with Xavier init. We apply AdamW optimization on a single GPU, set backbone learning rate to 2e-5, the learning rate of other parameters to 2e-4, and weight decay to 2e-4. The mini-batch size for each iteration is 25 image pairs. We set 120 epochs in total and each epoch have 1200 iterations. After 50 epochs, the learning rate has dropped by 10 times. Our approach is implemented in Python using PyTorch on a PC with Intel(R) Core (TM) i5-10400F CPU @ 2.90GHz 2.90 GHz, 16G RAM, Nvidia RTX 3060.

3.2. Comparison with the state-of-the-art
In this subsection, we compare our tracker with the recent state-of-the-art trackers on six tracking benchmarks which include GOT-10k, LaSOT, VOT2018, VOT2019, UAV123 and OTB-100.

3.2.1. GOT-10k.
Got-10k is a very large dataset, which contains more than 10000 videos for training and 180 for testing. This data set uses unique categories for tracking objects in the training set and test set. As shown in Figure 5, our tracker method is second only to the best tracker TransT, but better than other trackers.

3.2.2. LaSOT.
Lasot is a large-scale data set proposed recently. This data set consists of 1,400 challenging videos, including 1,120 training video and 280 test videos. We report the success (AUC) and precision (P and Pnorms) scores in Table 1. compared with state-of-the-art trackers, showing the superiority of our proposed method. Figure 6 shows the evaluation of different attributes on the LaSOT data set, indicating that CTFT has higher performance.

| SiamFC | MDNet | ECO | ATOM | SiamRPN++ | SiamBAN | SiamCAR | PrDiMP | KYS | Ocean | TransT | Ours |
|--------|-------|-----|------|-----------|---------|---------|--------|-----|-------|--------|------|
| Prec. (%) | 33.9  | 37.3 | 30.1 | 50.5 | 49.1 | 52.1 | 51.0 | 60.8 | - | 56.6 | 69.0 | 75.4 |
| N.Prec.(%) | 42.0 | 46.0 | 33.8 | 57.6 | 56.9 | 59.8 | 60.0 | 68.8 | 63.3 | 65.1 | 73.8 | 82.2 |
| Success (%) | 33.6 | 39.7 | 32.4 | 51.5 | 49.6 | 51.4 | 50.7 | 59.8 | 55.4 | 56.0 | 64.9 | 70.4 |
3.2.3. VOT2018 & VOT2019.
The VOT2018 and VOT2019 benchmark contains 60 challenging videos. According to the evaluation protocol, the tracker has a restart mechanism that can re-track after the tracking fails. The expected average overlap (EAO) is used to evaluate the performance of the dataset while considering both accuracy and robustness. In Table 2, we show the accuracy, robustness, and EAO scores of recent trackers with good performance. Our tracker has achieved the best performance on the vot2018 and vot2019 datasets respectively.

3.2.4. UAV123.
This challenging dataset contains 123 videos taken by UAV at low altitude, with the characteristics of small objects, fast motion, occlusion, absent, and distractor objects. As shown in Table 2, our tracker obtain 68.5% in terms of overall AUC score.

3.2.5. OTB-100.
OTB100 is a popular tracker containing 100 video sequences and 11 challenge attributes. In Figure 7, we compare results with state-of-the-art algorithms recently. Our CTFT achieves an AUC score of 75.1%, which is better than the most advanced methods available.

Table 2: Comparison with SOTA trackers on VOT2018 and VOT2019, with accuracy (A), robustness (R), and expected average overlap (EAO). Red, blue and green fonts indicate the top-3 trackers.

|           | VOT2018 |          |         |          |         |         |         |         |
|-----------|---------|----------|---------|----------|---------|---------|---------|---------|
| Tracked   | ATOM    | SiamRPN++| SiamFC++| DiMP     | PrDiMP  | SiamBAN | SiamAttn| Ocean   |
| A ↑       | 0.590   | 0.600    | 0.587   | 0.597    | 0.618   | 0.597   | 0.630   | 0.592   | 0.650   |
| R ↓       | 0.204   | 0.234    | 0.183   | 0.153    | 0.165   | 0.178   | 0.160   | 0.117   | 0.070   |
| EAO ↑     | 0.401   | 0.414    | 0.426   | 0.440    | 0.442   | 0.452   | 0.470   | 0.489   | 0.532   |
| VOT2019   |          |          |         |          |         |         |         |         |
| A ↑       | 0.603   | 0.599    | -       | -        | -       | 0.602   | -       | 0.594   | 0.648   |
| R ↓       | 0.411   | 0.482    | -       | -        | -       | 0.396   | -       | 0.316   | 0.231   |
| EAO ↑     | 0.292   | 0.285    | -       | -        | -       | 0.327   | -       | 0.350   | 0.355   |
| UAV123    |          |          |         |          |         |         |         |         |
| A ↑       | 0.650   | 0.613    | -       | 0.653    | 0.680   | 0.631   | 0.650   | -       | 0.685   |

Figure 7: Comparison with state-of-the-art methods on success and precision plots on OTB-100.

3.3. Ablation Study and Analysis

3.3.1. Fusion of Backbone layer.
Due to the structure of the swin transformer block, the backbone network can be constructed hierarchically. Like resnet50, swin transformer designs a network with obvious layers. The structure at the bottom handles more and more local data. The network at the top handles less data but has more
semantic information. Therefore, inspired by siamRPN++, we try to perform multi-layer fusion of the feature information extracted from different layers of the backbone. Unlike siamRPN++, we do not directly use linear weighting, but carry out adaptive fusion. As shown in Table 3, the layer feature fusion did not achieve the desired effect, so we explored the attention map of the tracking target after the fusion of different layers to explore the reason for the failure of the fusion in Figure 8. We think that the possible reason why the fusion does not improve the effect is that after the input sample image and the search area image are extracted from the backbone, the feature sizes obtained in the second stage, the third stage and the fourth stage are $32 \times 32$, $16 \times 16$, $8 \times 8$ (search area), $16 \times 16$, $8 \times 8$ and $4 \times 4$ (template area) respectively, as can be seen from table 3, the performance of the second and fourth phases of the test is very poor. In the second stage, the shallow layer of the network contains more simple features such as edge or location, which can not be tracked only by using simple features. In the fourth stage, the deep layer of the network contains more semantic information, but because of the low resolution, it can not be tracked only by using the fourth layer. After fusing the features of the second and fourth stages, it can be found that combining the shallow position information and the deep semantic information, the tracking effect will be better than the single one. After the fusion of second, third and fourth stage, because the resolution of the fourth layer is too small, it will cause some subtle effects on the third layer during the fusion, making the effect after fusion slightly worse than the third layer alone, so in this article, we choose the third layer as the input for subsequent feature fusion.

Table 3  Comparison with different layers on VOT2019 and OTB100, with accuracy (A), robustness (R), and expected average overlap (EAO). The best three results are shown in red, blue and green fonts.

|          | S2 | S3 | S4 | S23 | S24 | S34 | S234 | S3 online |
|----------|----|----|----|-----|-----|-----|------|-----------|
| VOT2019  | A↑ | 0.162 | 0.592 | 0.065 | 0.525 | 0.563 | 0.548 | 0.591 | 0.648 |
| R↓       | 6.993 | 0.567 | 7.444 | 0.767 | 0.767 | 0.958 | 0.652 | 0.231 |
| EAO↑     | 0.011 | 0.232 | 0.005 | 0.194 | 0.192 | 0.168 | 0.222 | 0.355 |
| OTB100   | A↑ | 0.025 | 0.661 | 0.040 | 0.538 | 0.626 | 0.601 | 0.655 | 0.751 |

Figure 8 Visualization of attention maps of different fusion layers.

3.3.2. Online fine tracking.
As shown in Table 3, after adding online tracking, the accuracy on the VOT2019 dataset has increased from 0.592 to 0.648, with an increase of 5.6 percentage points. At the same time, the robustness of the tracker has been greatly enhanced, and the robustness value decreases from 0.567 to 0.231. Our online tracking model further improves our tracker by 12.3 percentage points in terms of EAO, it achieves the best overall performance with the highest robustness and competitive accuracy. On the OTB100 dataset,
the online tracking model has also been improved by 9.0 percentage points, which shows that our model has strong adaptability and can cope with tracking in different situations.

4. Conclusion
In this work, we propose a simple and effective visual tracking framework. By replacing the original convolutional network backbone with the swin transformer block, we can obtain self-attention from global and long-range semantic information interaction, and try to explore the possibility of attention mechanism as the backbone. At the same time, online fine tracking is added, which greatly improves the performance of the tracker. Extensive experiments on six visual tracking benchmarks show that the proposed tracker has the most advanced performance, which proves the effectiveness of CTFT.

References
[1] Henriques, J. F. , Caseiro, R. , Martins, P. , & Batista, J. . (2015). High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis & Machine Intelligence, 37(3), 583-596.
[2] Held, D. , Thrun, S. , & Savarese, S. . (2016). Learning to Track at 100 FPS with Deep Regression Networks. European Conference on Computer Vision. Springer, Cham.
[3] Kalal, Z. , Mikolajczyk, K. , & Matas, J. . (2011). Tracking-learning-detection. IEEE Transactions on Software Engineering, 34(7), 1409-1422.
[4] Yi, W. , Lim, J. , & Yang, M. . (2013). Online object tracking: a benchmark supplemental material. IEEE.
[5] Bertinetto, L. , Valmadre, J. , Henriques, J. F. , Vedaldi, A. , & Torr, P. . (2016). Fully-convolutional siamese networks for object tracking. European Conference on Computer Vision.
[6] Bo, L. , Yan, J. , Wei, W. , Zheng, Z. , & Hu, X. . (2018). High Performance Visual Tracking with Siamese Region Proposal Network. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.
[7] Wang, Q. , Teng, Z. , Xing, J. , Gao, J. , & Maybank, S. . (2018). Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE.
[8] Wang, Q. , Zhang, M. , Xing, J. , Gao, J. , & Maybank, S. . (2018). Do not lose the details: reinforced representation learning for high performance visual tracking. Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18.
[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., 2017.
[10] Carion, N. , Massa, F. , Synnaeve, G. , Usunier, N. , Kirillov, A. , & Zagoruyko, S. . (2020). End-to-End Object Detection with Transformers.
[11] 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,” CoRR, vol. abs/2103.14030, 2021.
[12] Chen, X. , Yan, B. , Zhu, J. , Wang, D. , Yang, X. , and Lu, H. , “Transformer Tracking”, 2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.
[13] Pinheiro, P. O. , Collobert, R. , and Dollar, P. , “Learning to Segment Object Candidates”, 2015.
[14] H. Rezatofighi, T. Soi, N. , JY Gwak, Sadeghian, A. , & Savarese, S. . (2019). Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.