Robust Object Modeling for Visual Tracking

Yidong Cai  Jie Liu*  Jie Tang  Gangshan Wu
State Key Laboratory for Novel Software Technology, Nanjing University, China
yidong_cai@smail.nju.edu.cn; {liujie, tangjie, gswu}@nju.edu.cn

Abstract

Object modeling has become a core part of recent tracking frameworks. Current popular tackers use Transformer attention to extract the template feature separately or interactively with the search region. However, separate template learning lacks communication between the template and search regions, which brings difficulty in extracting discriminative target-oriented features. On the other hand, interactive template learning produces hybrid template features, which may introduce potential distractors to the template via the cluttered search regions. To enjoy the merits of both methods, we propose a robust object modeling framework for visual tracking (ROMTrack), which simultaneously models the inherent template and the hybrid template features. As a result, harmful distractors can be suppressed by combining the inherent features of target objects with search regions’ guidance. Target-related features can also be extracted using the hybrid template, thus resulting in a more robust object modeling framework. To further enhance robustness, we present novel variation tokens to depict the ever-changing appearance of target objects. Variation tokens are adaptable to object deformation and appearance variations, which can boost overall performance with negligible computation. Experiments show that our ROMTrack sets a new state-of-the-art on multiple benchmarks.

1. Introduction

Visual object tracking (VOT) [1, 4, 8, 12, 22, 25, 35, 39, 41, 47, 57, 60] is a fundamental task in computer vision, which aims at localizing an arbitrary target in video sequences given its initial status. The occlusion, scale variation, object deformation, and co-occurrence of distractor objects pose a challenge to acquiring an effective tracker in real-world scenarios. Current dominating trackers typically address these problems with a Transformer-based [42] architecture.

*Corresponding author.

The core components in a typical Transformer tracking framework are the object modeling blocks. As demonstrated in Figure 1(a), the two-stream hybrid modeling methods [6, 50] learn the template feature interactively with the search region via two cross-attention (CA) operations. Instead of cross-attention, the one-stream hybrid modeling methods [5, 54] in Figure 1(c) jointly learn the hybrid template feature and search region feature with one self-attention (SA) operation. Different from hybrid template modeling, the two-stream separate modeling [10, 23] in Figure 1(b) keeps an inherent template stream to ensure the purity of template features. Separate template learning can keep the inherent features of target templates, which prevents interference from search regions. Though suffering from potential distractors, hybrid template learning conducts extensive feature matching between the template and search region, thus allowing mutual guidance for target-oriented feature extraction.

In order to enjoy the merits of separate and hybrid template modeling simultaneously, we propose a robust ob-
ject modeling framework for visual tracking (named ROMTrack). As shown in Figure 1(d), our robust modeling scheme involves two kinds of templates, the inherent template \( it \) and the hybrid template \( ht \). Meanwhile, our scheme also designs the novel variation tokens \( vt \). The inherent template applies self-attention separately to enhance its learned feature. Besides, it accepts queries from the hybrid template and the search region features to provide inherent information for discriminative target-oriented feature learning. The bottom part of Figure 1(d) is a hybrid attention that adopts a standard cross-attention operation to enhance the template and search region features with mutual guidance. Furthermore, it is well-recognized that tracking is a task suffering from object deformation and appearance variations \([10, 57]\). We tackle this problem by introducing novel variation tokens to improve robustness. It is observed that the target’s motion during a short period is usually smooth but may be accompanied by large changes in appearance \([6, 58]\). The tracker can easily handle smooth motion, but appearance changes are hard to distinguish. Therefore, we generate variation tokens from hybrid template features to leverage appearance information during tracking. Despite the simplicity, our variation tokens perform well with negligible computation.

The main contributions of this work are three-fold:

- We propose a robust object modeling framework for visual tracking (ROMTrack). It can keep the inherent information of the target template and enables mutual feature matching between the target and the search region simultaneously.
- We present a neat and effective variation-token design that embeds the object’s appearance context during tracking into the attention calculation of hybrid target-search features.
- The proposed ROMTrack sets a new state-of-the-art performance on six challenging benchmarks, including GOT-10k \([26]\), LaSOT \([18]\), TrackingNet \([38]\), LaSOText \([17]\), OTB100 \([49]\), and NFS30 \([21]\).

2. Related Work

In this section, we briefly review different visual object tracking methods and the Transformer attention mechanism in general vision tasks.

Visual Object Tracking. Early Siamese-based trackers \([2, 7, 16, 19, 29, 30, 45, 51, 56, 58]\) first extract the template and search region features separately by a CNN (Convolutional Neural Network) backbone with shared structure and parameters. Then, a correlation-based network is responsible for computing the similarity between the template and the search region. Correlation modeling plays a critical role in tracking networks. However, conventional correlation-based networks do not fully use the global context. Therefore, recent dominating trackers \([6, 9, 10, 20, 23, 31, 44, 53, 54, 55]\) introduce stacked Transformer layers for better relation modeling.

The pioneering Transformer tracking method TransT \([6]\) adopts a similar pipeline as Siamese-based trackers, where the lightweight relation modeling network is replaced with relatively heavy Transformer layers. The two-stream attention in TransT enables bi-directional information interaction. Unlike TransT, MixFormer \([10]\) utilizes the flexibility of attention operations for simultaneous feature extraction and relation modeling. MixFormer also adopts a two-stream attention pipeline but prunes the cross-attention from the target’s query to the search area, eliminating potential negative influence from distractors. AiATrack \([23]\) employs a similar asymmetric scheme, where the search region conducts queries on the target feature while the target only enhances its feature with self-attention blocks. In order to bridge a free information flow between the template and search region, OSTrack \([54]\) adopts a one-stream attention scheme. It concatenates the flattened template and search region and feeds them into stacked self-attention layers for joint feature learning and relation modeling. However, the extensive feature fusion of self-attention layers may bring interfering information to the target feature due to potential distractors. Instead, we propose a robust object modeling scheme that contains an inherent template stream, a variation-token stream, and a bi-directional template-search stream, leading to a more accurate transformer tracker.

Transformer Attention. The attention mechanism \([42]\) has played an increasingly important role in computer vision in the past few years. And recently, in most vision tasks, attention architectures represented by Transformer have obtained impressive performances. To be more specific, Transformer attention is usually helpful for modeling spatial features and temporal relations. For example, the Vision Transformer (ViT) \([15]\) and other following works, including PVT \([46]\), CVT \([48]\), and Swin-Transformer \([33]\) have shown their capacity to aggregate spatial information and benefit many downstream tasks. Transformer attention has also been utilized in visual tracking but has yet to be fully exploited. Most of them focus on designing complex structures. Instead, in this work, we try to explain the potential defects of previous trackers and seek an approach for robust object modeling. We follow the pure Transformer architecture to further explore attention mechanism for visual tracking.

3. Method

We propose ROMTrack, a robust object modeling network for tracking, in Figure 2. We first give an overview of the proposed ROMTrack architecture and then elaborate on the proposed object encoder. Finally, we give a discussion
on our modeling method.

### 3.1. Overall Architecture

**Backbone.** As shown in Figure 2(a), we adopt the vanilla ViT [15] as the backbone. More concretely, we replace the conventional ViT encoder with the proposed object encoder and add a prediction head on the output tokens of the last encoder. The input of ROMTrack is a triplet of images containing a template image pair \((it_{img}, ht_{img})\) \(\in \mathbb{R}^{3 \times H_x \times W_z}\) and one search region image \(sr_{img} \in \mathbb{R}^{3 \times H_{sr} \times W_{sr}}\). The \(it_{img}\) is responsible for learning inherent template features and \(ht_{img}\) is accountable for learning hybrid template features. Following ViT [15], we split all input images and flatten them into sequences of patches: \(it \in \mathbb{R}^{N_t \times 3 \times P^2}\), \(ht \in \mathbb{R}^{N_t \times 3 \times P^2}\), and \(sr \in \mathbb{R}^{N_{sr} \times 3 \times P^2}\), where \(P \times P\) is the resolution of each patch, \(N_t = H_t W_t / P^2\) and \(N_{sr} = H_{sr} W_{sr} / P^2\) are the number of patches of the templates and the search region, respectively. Then we generate \(D\)-dimensional patch embeddings using a linear projection layer. After adding position embeddings, the resulting token sequences are ready for \(N\) stacked object encoders. The encoder layer employs robust object modeling to learn discriminative feature representations, which will be elaborated in Section 3.2.

**Prediction Head.** As pointed out in previous work [10], corner-based [28] localization heads may have a bad effect on the modeling capacity of deeper transformer encoders. Consequently, we adopt a fully convolutional center-based [59] localization head to estimate the bounding box of tracked objects, which consists of \(L\) stacked Conv-BN-ReLU layers. Specifically, the target classification score map \(C \in [0, 1]^{H_{sr} \times W_{sr}}\), the local offset map \(O \in [0, 1]^{2 \times H_{sr} / P \times W_{sr} / P}\), and the normalized bounding box size map \(S \in [0, 1]^{H_{sr} / P \times W_{sr} / P}\) are generated by the center head. Finally, the position with the highest classification score in \(C\) is considered the target position and the target bounding box can be calculated using \(O\) and \(S\).

The classification branch is supervised using Gaussian weighted focal loss [28] during training. Specifically, given a ground truth target center \(\hat{c}\) and the corresponding position \(\hat{c} = [\hat{x}, \hat{y}]\) in feature map, the ground truth heatmap can be formulated as \(\hat{C}_{xy} = e^{-\frac{(x-\hat{x})^2+(y-\hat{y})^2}{2\sigma^2}}\), where \(\sigma\) is a standard deviation adaptive to object size. So the Gaussian weighted focal loss is employed as follows:

\[
L_{cls} = -\sum_{xy} \mathbb{I}(\hat{C}_{xy} = 1)(1 - C_{xy})^{\alpha} \log(C_{xy}) + (1 - \hat{C}_{xy})^{\beta}(C_{xy})^{\alpha} \log(1 - C_{xy}),
\]

where \(\mathbb{I}(\cdot)\) is the indicator function, \(\alpha\) and \(\beta\) are hyper-parameters, and we set them to 2 and 4 following [28, 54].

As for the bounding box regression branch, \(L_1\) loss and \(GIoU\) loss are adopted. Generally, We set different weights for different losses: \(\lambda_{L_1} = 5\), \(\lambda_{giou} = 2\) and \(\lambda_{cls} = 1\). And both training stages share the same loss function as follows:

\[
L_{total} = \lambda_{L_1} L_1 + \lambda_{giou} L_{giou} + \lambda_{cls} L_{cls}.
\]

### 3.2. Object Encoder

The proposed object encoder in Figure 2(b) contains two critical components, i.e., variation tokens and robust object modeling. Before describing the principles of robust object modeling, we first explain our design of variation tokens.
Variation Tokens. Variation tokens are the embedding of contextual appearance changes of the target object, which helps to tackle the problem of object deformation and appearance variations. As shown in Figure 3, the variation tokens $v_{t}$ are generated after each object encoder and encode the variation of appearance context from the search region, which will be further demonstrated later. The generation and usage of variation tokens can be formulated as follows:

$$v_{t,k,t} = h_{t,k,t-1},$$  \hfill (3)

$$F_{k+1} = \text{ObjectEncoder}_{k+1} \left( \text{Concat} (v_{t,k,t}, F_{k}^{t}) \right),$$  \hfill (4)

where $F$ represents the output features, $k$ is the encoder index and $t$ denotes the $t$-th frame. So $F_{k}^{t}$ is the output features of $k$-th encoder in frame $I_{t}$, and $h_{t,k,t}$ is the hybrid template part of $F_{k}^{t}$, which incorporates appearance information from the search region.

Equation 3 indicates that we reserve hybrid template tokens of frame $I_{t-1}$ as the input to frame $I_{t}$, because appearance variations of the target object in $I_{t}$ relative to $I_{t-1}$ are encoded at the feature level of these tokens. Furthermore, Equation 4 aims to embed the variation of object appearance into the network. Specifically, it feeds the variation tokens $v_{t,k,t}$ together with the output features $F_{k}^{t}$ to the $(k+1)$-th encoder when tracking the $t$-th frame. The MACs are negligible because the construction of variation tokens only includes embedding assignments (Equation 3). Meanwhile, the employment of variation tokens is just a combination of token concatenation and a series of lightweight cross-attention operations related to $v_{t,k,t}$ (Equation 4).

Robust Object Modeling. One-stream hybrid modeling enables extensive bi-directional information flows between the template-search image pairs, and discriminative target-oriented features can be dynamically extracted by mutual guidance. However, excessive communications may suffer from tracking failures and background clutters. Two-stream separate modeling can keep a separate template stream to avoid negative influences from potential distractors, but its extracted template features are inadequate to object deformations and appearance changes.

To address these problems, we propose a robust object modeling method. As shown in Figure 2(b), the input of object attention consists of four parts, i.e., the inherent template $i_{t} \in \mathbb{R}^{N_{t} \times D}$, the hybrid template $h_{t} \in \mathbb{R}^{N_{t} \times D}$, the search region $sr \in \mathbb{R}^{N_{sr} \times D}$, and the variation tokens $vt \in \mathbb{R}^{N_{vt} \times D}$. Following the conventional attention block, we use a linear projection layer to produce the d-dimensional (query, key, value) triplet. For example, the triplet for $i_{t}$ is $(q_{dt}, k_{it}, v_{it})$. Then we conduct self-attention on $i_{t}$ to learn pure template features:

$$A_{it} = \text{Softmax} \left( \frac{q_{it}k_{it}^{T}}{\sqrt{d}} \right)v_{it},$$  \hfill (5)

where $A_{it}$ represents the output of the self-attention operation. The inherent template feature is enhanced through self-attention, eliminating interference from the search region. Meanwhile, the hybrid template feature and the search region feature are learned via a cross-attention operation. Let $(q_{z}, k_{z}, v_{z})$ denote the $(query, key, value)$ triplet of the cross-attention, where $q_{z}$, $k_{z}$, and $v_{z}$ are defined as follows:

$$q_{z} = [q_{ht}, q_{sr}],$$  \hfill (6)

$$k_{z} = [k_{vt}, k_{it}, k_{ht}, k_{sr}],$$  \hfill (7)

$$v_{z} = [v_{vt}, v_{it}, h_{ht}, h_{sr}].$$  \hfill (8)

Equation 6 - 8 show that the inputs of cross-attention are a rearrangement and concatenation (denoted by $[\ldots]$) of the search region features (indicated by subscript $sr$), the two types of template features (indexed by subscript $ht$ and $it$), and the variation token features (indicated by subscript $vt$). The output of cross-attention operation $A_{z}$ can be obtained via:

$$A_{z} = \text{Softmax} \left( \frac{q_{z}k_{z}^{T}}{\sqrt{d}} \right)v_{z},$$  \hfill (9)

The hybrid template feature and search region feature inside $A_{z}$ get enhanced by fusing informative features from the inherent template and variation tokens. As a result, the network is able to obtain the information of both the original target (i.e., object in the first frame) and the ever-changing target (i.e., object in the $(t-1)$-th frame) when tracking the $t$-th frame.

For further explanation, we conduct a more in-depth analysis below. Let $M_{z}$ be the correlation map calculated in the cross-attention, then $M_{z}$ can be written as:

$$M_{z} = \text{Softmax} \left( \frac{q_{z}k_{z}^{T}}{\sqrt{d}} \right)$$

$$= \text{Softmax} \left( \frac{q_{ht}k_{vt}^{T} q_{vt}k_{ht}^{T} q_{ht}k_{sr}^{T} q_{sr}k_{ht}^{T} q_{sr}k_{vt}^{T}}{\sqrt{d}} \right)$$

$$= \begin{bmatrix} M_{ht,vt} M_{ht,it} M_{ht,ht} M_{ht,sr} \\ M_{sr,vt} M_{sr,it} M_{sr,ht} M_{sr,sr} \end{bmatrix},$$  \hfill (10)
where $M_{a,b}$ is a measure of similarity between $a$ and $b$, e.g., $M_{ht, sr}$ refers to the similarity between the hybrid template and search region. Based on Equation 10, the attention output $A_z$ can be rewritten as:

$$A_z = M_z v_z = \text{Softmax} \left( \frac{q_k k^T}{\sqrt{d}} \right) v_z$$

$$= \begin{bmatrix} M_{ht,vt} & M_{ht,it} & M_{ht,ht} & M_{ht,sr} \\ M_{sr,vt} & M_{sr,it} & M_{sr,ht} & M_{sr,sr} \end{bmatrix} \begin{bmatrix} v_{vt} \\ v_{it} \\ v_{ht} \\ v_{sr} \end{bmatrix} (11)$$

$$= \begin{bmatrix} M_{ht,vt} v_{vt} + M_{ht,it} v_{it} + M_{ht,ht} v_{ht} + M_{ht,sr} v_{sr} \\ M_{sr,vt} v_{vt} + M_{sr,it} v_{it} + M_{sr,ht} v_{ht} + M_{sr,sr} v_{sr} \end{bmatrix} \triangleq \begin{bmatrix} A_{ht} \\ A_{sr} \end{bmatrix},$$

where $A_{ht}$ and $A_{sr}$ denote the generated hybrid template and search region features, respectively. It is easy to figure out that both $A_{ht}$ and $A_{sr}$ aggregate information from the inherent template (e.g., $M_{ht,vt} v_{vt}$ and $M_{sr,vt} v_{vt}$) and variation tokens (e.g., $M_{ht,it} v_{it}$ and $M_{sr,it} v_{it}$) to enhance their features.

Recall that in Figure 3, we use the hybrid template $ht_{k,t-1}$ to generate the variation tokens $vt_{k,t}$ to provide variation information of contextual appearance for the next frame. This is reasonable because the $M_{ht,vt} v_{vt}$ term in $A_{ht}$ has incorporated the feature of search region into the hybrid template, helping the output hybrid template tokens to capture current information of the search region. In other words, the appearance information of the target object in both the first frame and the current frame is incorporated into the hybrid template tokens, making them sensitive to contextual appearance changes.

Therefore, we can cache the hybrid template as variation tokens in the next frame to leverage appearance information during tracking. Overall, with the variation-token design, the feature extraction and information integration process are unified in the proposed object modeling framework.

### 3.3. Discussions

**Necessity of Hybrid Template.** The hybrid template serves two primary purposes. The first is to conduct extensive feature matching between the template and search region, thus allowing mutual guidance for target-oriented feature extraction. The second is to encode the variation of appearance context by interacting with the search region, which helps variation tokens model appearance changes of objects between adjacent frames. Further analysis is conducted in Section 4.3.

**Training and Inference.** The training process contains two stages. In the first stage, we follow the standard training recipe of mainstream trackers [6, 10, 53] to train our ROMtrack without variation tokens, i.e., only with the inherent and hybrid templates. In the second stage, we add variation tokens into training by sampling two search regions in consecutive frames of the same sequence to model the appearance variations between them. For inference, only the initial template and the cropped search region are fed into the ROMTrack pipeline to produce the target bounding box. The initial template serves as the input for both inherent and hybrid templates. During the tracking procedure, the variation tokens are obtained per frame and employed for subsequent tracking.

### 4. Experiments

#### 4.1. Implementation Details

Our trackers are implemented using Python 3.6.13 and PyTorch 1.7.1. The models are trained on 8 Tesla V100 GPUs, and we test the inference speed on a single NVIDIA1080Ti GPU.

**Model.** We adopt the vanilla ViT-Base [15] model pre-trained with MAE [24] on ImageNet [14] as the backbone of our ROMTrack. All the input images are split into 16×16 patches. As for the prediction head, we adopt a lightweight FCN consisting of 4 stacked Conv-BN-ReLU layers for each output. To build an efficient tracker, we adopt a smaller image resolution than other trackers [10, 23, 53]. Namely, the sizes of the template and search images are 128×128 pixels and 256×256 pixels, respectively. Furthermore, to verify the scalability of our proposed ROMTrack, we also provide an implementation with a higher resolution called ROMTrack-384, and the sizes of the template and search images are 192×192 pixels and 384×384 pixels.

**Training.** The training splits of COCO [32], GOT-10k [26], LaSOT [18], and TrackingNet [38] are used for training. While for the GOT-10k test, we follow the one-shot protocol by only using the GOT-10k train split for training. The training process of ROMTrack consists of two stages: the first 300 epochs are for the backbone and head, and the extra 100 are to merge the variation tokens into our architecture. For data augmentations, horizontal flip and brightness jittering are used following the convention [10, 53, 54]. We train the ROMTrack using AdamW [34] with weight decay set to $10^{-4}$. For the first stage, the learning rate is initialized as $4 \times 10^{-4}$ and decreased to $4 \times 10^{-5}$ at the epoch of 240. For the second stage, the learning rate is initialized as $4 \times 10^{-5}$ and decreased to $4 \times 10^{-6}$ at the epoch of 80.

**Inference.** We adopt the Hanning window penalty to utilize positional prior in tracking following the common practice [6, 54, 58]. To be more specific, the classification map $C$ is multiplied by the Hanning window with the same size to generate confidence scores, and we simply select the prediction box with the highest confidence score as result.
Table 1: Comparison with state-of-the-art on four large-scale benchmarks: GOT-10k, LaSOT, TrackingNet, LaSOT$_\text{ext}$. The best two results are shown in red and blue fonts. * denotes the model trained with only GOT-10k train split.

<table>
<thead>
<tr>
<th>Method</th>
<th>Source</th>
<th>GOT-10k*</th>
<th>LaSOT</th>
<th>TrackingNet</th>
<th>LaSOT$_\text{ext}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROMTrack</td>
<td>Ours</td>
<td>72.9</td>
<td>82.9</td>
<td>70.2</td>
<td>69.2</td>
</tr>
<tr>
<td>SwinTrack-T-224</td>
<td>NIPS22</td>
<td>71.3</td>
<td>81.9</td>
<td>64.5</td>
<td>67.2</td>
</tr>
<tr>
<td>OSTDtrack-256</td>
<td>ECCV22</td>
<td>71.0</td>
<td>80.4</td>
<td>68.2</td>
<td>69.1</td>
</tr>
<tr>
<td>OSTDtrack-256(w/o CE)</td>
<td>[54]</td>
<td>71.0</td>
<td>80.3</td>
<td>68.2</td>
<td>68.7</td>
</tr>
<tr>
<td>AiATrack</td>
<td>ECCV22</td>
<td>69.6</td>
<td>80.0</td>
<td>63.2</td>
<td>69.0</td>
</tr>
<tr>
<td>SimTrack-B16</td>
<td>[65]</td>
<td>68.6</td>
<td>78.9</td>
<td>62.4</td>
<td>69.3</td>
</tr>
<tr>
<td>UnicorN</td>
<td>[52]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MxFormer-22k[10]</td>
<td>CVPR22</td>
<td>70.7</td>
<td>80.0</td>
<td>67.8</td>
<td>69.2</td>
</tr>
<tr>
<td>MxFormer-4k[10]</td>
<td>CVPR22</td>
<td>71.2</td>
<td>79.9</td>
<td>62.3</td>
<td>67.9</td>
</tr>
<tr>
<td>ToMPSP0</td>
<td>[36]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ToMP101[36]</td>
<td>CVPR22</td>
<td>72.3</td>
<td>80.0</td>
<td>72.2</td>
<td>67.6</td>
</tr>
<tr>
<td>SBT-large[59]</td>
<td>CVPR22</td>
<td>70.4</td>
<td>80.8</td>
<td>64.7</td>
<td>66.7</td>
</tr>
<tr>
<td>KeepTrack</td>
<td>ICCV21</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STARK[53]</td>
<td>ICCV21</td>
<td>68.8</td>
<td>78.1</td>
<td>64.1</td>
<td>67.1</td>
</tr>
<tr>
<td>DTT[55]</td>
<td>ICCV21</td>
<td>63.4</td>
<td>74.9</td>
<td>51.4</td>
<td>60.1</td>
</tr>
<tr>
<td>TDT[56]</td>
<td>CVPR21</td>
<td>67.1</td>
<td>76.8</td>
<td>60.9</td>
<td>64.9</td>
</tr>
<tr>
<td>TDiDMP[44]</td>
<td>CVPR21</td>
<td>67.1</td>
<td>77.7</td>
<td>58.3</td>
<td>63.9</td>
</tr>
<tr>
<td>LTMU[11]</td>
<td>CVPR20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SiamR-CNN</td>
<td>CVPR20</td>
<td>64.9</td>
<td>72.8</td>
<td>59.7</td>
<td>64.8</td>
</tr>
<tr>
<td>Ocean[10]</td>
<td>ECCV20</td>
<td>61.1</td>
<td>72.1</td>
<td>47.3</td>
<td>56.0</td>
</tr>
<tr>
<td>D3MP[3]</td>
<td>ICCV19</td>
<td>61.1</td>
<td>71.7</td>
<td>49.2</td>
<td>56.9</td>
</tr>
<tr>
<td>SiamRPN++</td>
<td>[29]</td>
<td>57.1</td>
<td>61.6</td>
<td>32.5</td>
<td>49.6</td>
</tr>
<tr>
<td>MDNet[59]</td>
<td>CVPR16</td>
<td>29.9</td>
<td>30.3</td>
<td>9.9</td>
<td>39.7</td>
</tr>
<tr>
<td>SiamFC[12]</td>
<td>ECC16</td>
<td>34.8</td>
<td>35.3</td>
<td>9.8</td>
<td>33.7</td>
</tr>
</tbody>
</table>

Table 2: Comparison with state-of-the-art trackers on two small-scale benchmarks: OTB100 and NFS30. Results are compared in terms of AUC(%) score. The best two results are shown in red and blue fonts.

<table>
<thead>
<tr>
<th>Method</th>
<th>OTB100</th>
<th>NFS30</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiamRPN++</td>
<td>69.6</td>
<td>50.3</td>
</tr>
<tr>
<td>PrDiMP</td>
<td>69.8</td>
<td>64.8</td>
</tr>
<tr>
<td>SuperDiMP</td>
<td>68.4</td>
<td>65.7</td>
</tr>
<tr>
<td>TransT</td>
<td>65.2</td>
<td>65.2</td>
</tr>
<tr>
<td>STARK</td>
<td>70.5</td>
<td>66.4</td>
</tr>
<tr>
<td>KeepTrack</td>
<td>70.0</td>
<td>66.5</td>
</tr>
<tr>
<td>RTS</td>
<td>70.4</td>
<td>66.7</td>
</tr>
<tr>
<td>ToMP</td>
<td>67.1</td>
<td>66.5</td>
</tr>
<tr>
<td>MxFormer-L[10]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OSTDtrack-384</td>
<td>70.9</td>
<td>67.9</td>
</tr>
<tr>
<td>OSTrack-384</td>
<td>71.4</td>
<td>68.0</td>
</tr>
</tbody>
</table>

4.2. Comparison with State-of-the-Art Trackers

We compare our ROMTrack with state-of-the-art (SOTA) trackers on six different benchmarks, including four well-known large-scale benchmarks and two commonly used small-scale benchmarks. Results on other datasets are available in Appendix.

GOT-10k. GOT-10k [26] is a large-scale dataset containing more than 10000 video segments of real-world moving objects. The object classes between train and test sets are zero-overlapped. We follow the one-shot protocol to only train our model on the GOT-10k training split and evaluate the results through the evaluation server. As presented in Table 1, ROMTrack improves all metrics by a large margin, e.g., 1.6% in AO compared with SwinTrack-T-224 and 2% in SR$_0.75$ compared with OSTDtrack-256, which indicates the capability in accurate discrimination and localization of objects. Furthermore, our higher resolution model ROMTrack-384 sets a new SOTA on the GOT-10k test split, demonstrating that our method has excellent potential to track objects of unseen classes by robust object modeling.

LaSOT. LaSOT [18] is a large-scale, long-term tracking benchmark containing 1400 video sequences: 1120 for training and 280 for testing. We evaluate our ROMTrack on the test set to compare with previous SOTA trackers. As reported in Table 1, our ROMTrack shows more accurate and balanced performance, surpassing both OSTDtrack and MxFormer in all three metrics. Specifically, our higher resolution model ROMTrack-384 establishes a new state-of-the-art on AUC of 71.4%. The result demonstrates that our approach benefits the long-term tracking scenarios. More analysis of the performance improvements on the LaSOT dataset can be found in Appendix.

TrackingNet. TrackingNet [38] is a large-scale short-term tracking benchmark that provides more than 30000 video sequences with over 14 million boxes. The test split of TrackingNet contains 511 sequences without publicly available ground truth and covers diverse target classes and scenes. We submit the tracking results to the official eval-
Table 3: Comparison of inference speed, MACs, and Params. We include the results of OStrack without candidate elimination (w/o CE) here for a fair speed comparison. * denotes the model trained with only GOT-10k train split.
Table 4: (a) Architectures of OSTrack-256 (w/o CE), STM, HTM and our ROMTrack (w/o vt). (b) Ablation study on inherent template and hybrid template. The best results are in **bold** fonts. * denotes the model trained with only GOT-10k train split.

Table 5: Ablation study on variation tokens and exploration of template updating. The best results are in **bold** fonts.

Table 6: Ablation study on aligned comparison and sampling strategy. RS and CS denote random sampling and consecutive sampling. The best results are in **bold** fonts.

It is concluded that our variation-token design is vital to improve the overall performance of trackers. With the assistance of a template updating strategy, the performance can be further boosted, indicating that our variation tokens work complementary with the template update strategy.

**Study on Aligned Comparison.** The training of our tracker consists of two stages, adding up to 400 epochs in total. To prove that the outstanding performance of our tracker is not due to a longer training process, we select the HTM approach and train it for 400 epochs, denoted as HTM-400. The results are shown in the first two rows of Table 6. Our method still outperforms HTM-400 by a large margin (e.g., +0.5% AUC on LaSOT and +2.6% AUC on LaSOT\_ext), which proves that a simple extension of the training process is not the critical factor for excellent performance. Therefore, the robust object modeling approach is undoubtedly helpful for feature extraction and relation modeling.

**Study on Sampling Strategy.** During the second training stage, we employ a particular sampling strategy called consecutive sampling (CS). Different from random sampling (RS), two consecutive frames instead of two random frames in the same video sequence are sampled as the search region. The training process is described in Section 3.3. The results in Table 6 show that the consecutive sampling strategy obtains better performance because it helps the model to learn more about the temporal variation of object appearance. In addition, the model trained with random sampling also performs better than HTM, indicating the effectiveness of our robust object modeling.

**Visualizations.** To explore how the robust object modeling method works in our framework, we also visualize some
attention maps and search region features in Figure 4. We observe that:

- The object in the search region is enhanced layer by layer through interaction with the two template streams and the variation tokens.
- Possible distractors in the background get suppressed with our ROMTrack (Row 1, Row 2, and Row 5), suggesting the robustness of our method.
- Both HTM (Row 3) and STM (Row 4) have difficulty in distinguishing distractors with target objects while our ROMTrack locates objects more accurately.

As a result, our ROMTrack shows excellent tracking performance.

5. Conclusions

This work proposes a robust object modeling framework for visual tracking (ROMTrack). The proposed ROMTrack utilizes two template streams to learn robust and discriminative feature representations. The inherent template keeps original features of target objects, and the hybrid template learns mixed template-search features. The hybrid template can extract helpful information from the inherent template to form target-oriented features. Besides, the variation tokens are introduced to embed appearance context, thus adaptable to object deformation and appearance variations. The variation-token design is subtly integrated into attention computation, leading to a neat and effective tracker. Extensive experiments show that ROMTrack performs better than previous methods on multiple benchmarks.

Acknowledgements. This work is supported by National Key R&D Program of China (No. 2022ZD0160900), National Natural Science Foundation of China (No. 62076119, No. 61921006, No. 62072232), Fundamental Research Funds for the Central Universities (No. 020214380091, No. 020214380099), and Collaborative Innovation Center of Novel Software Technology and Industrialization.
References


