Effective Local and Global Search for Fast Long-term Tracking

Haojie Zhao, Bin Yan, Dong Wang, Xuesheng Qian, Xiaoyun Yang, Huchuan Lu

Abstract—Compared with short-term tracking, long-term tracking remains a challenging task that usually requires the tracking algorithm to track targets within a local region and re-detect targets over the entire image. However, few works have been done and their performances have also been limited. In this paper, we present a novel robust and real-time long-term tracking framework based on the proposed local search module and re-detection module. The local search module consists of an effective bounding box regressor to generate a series of candidate proposals and a target verifier to infer the optimal candidate with its confidence score. For local search, we design a long short-term updated scheme to improve the target verifier. The verification capability of the tracker can be improved by using several templates updated at different times. Based on the verification scores, our tracker determines whether the tracked object is present or absent and then chooses the tracking strategies of local or global search, respectively, in the next frame. For global re-detection, we develop a novel re-detection module that can estimate the target position and target size for a given base tracker. We conduct a series of experiments to demonstrate that this module can be flexibly integrated into many other tracking algorithms for long-term tracking and that it can improve long-term tracking performance effectively. Numerous experiments and discussions are conducted on several popular tracking datasets, including VOT, OxUvA, TLP, and LaSOT. The experimental results demonstrate that the proposed tracker achieves satisfactory performance with a real-time speed.

Index Terms—Visual object tracking, long-term tracking, global re-detection

1 Introduction

Visual object tracking is a fundamental task in computer vision and plays an important role in many realistic applications, such as surveillance, robotics, augmented reality, and unmanned drones [1], [2]. During online tracking, trackers are required to follow a generic object in consecutive video frames. However, this approach comes with several challenges, including occlusion, illumination variation, viewpoint change, rotation, and motion blur. With the help of deep learning and correlation filter methods, great successes [3], [4], [5], [6], [7], [8], [9] have been done in recent years. Numerous trackers have achieved satisfactory performance on popular tracking benchmarks [10], [11], [12].

With the rapid development of short-term tracking algorithms, an increasing number of researchers have focused on the long-term tracking task. Several long-term tracking benchmarks (e.g., VOT2018-LT [13], and OxUvA [14]) and trackers (e.g., [15], [16]) have been recently proposed. Compared with short-term tracking, long-term tracking has two salient features. First, trackers are required to track targets in a long-term video with thousands of frames, which is considerably longer than a short-term video. For example, the average frame length of 60 videos in the VOT2017 dataset [12] is approximately 356. By contrast, the average length of 35 videos in the VOT2018-LT dataset is more than 4100 frames, which is approximately 11 times longer than that of the VOT short-term dataset. It brings great challenge to previous tracking algorithms. Second, the tracked object in a long-term video may disappear frequently. For example, each sequence of the VOT2018-LT dataset contains on average 12 target disappearances, whereas that of UAV20L [17] is only 2. This difference between long-term and short-term tracking is crucial.

Some long-term trackers have been developed based on hand-crafted features, including TLD [18], LCT [19], FuCoLoT [20], MUSTer [21], CMT [22], and EBT [23]. However, they cannot achieve satisfactory performance on recent long-term benchmarks (see experiments in [14], [24]). Recent deep-learning-based long-term algorithms [24], [25], [26] have significantly improved the tracking performance. However, a robust and real-time framework for long-term tracking remains lacking. To fill in this gap, we propose a novel local-global search framework for long-term tracking. The local search module aims to precisely capture the tracked object in a local search region; meanwhile, the global search module focuses on efficiently predicting a suitable local search region for the base tracker and estimates the target position and the target size by generating a reference box.

In long-term tracking, the target may disappear for a long period time and reappear at an arbitrary position with an arbitrary size. This phenomenon undermines the assumptions of motion continuity and scales change continuity in the short-term tracking setting [27], [28]. In general, a short-term tracker locates the target object within a local search region. This search region is cropped around the previous target position and is several times the previous target size. That is, local tracking depends on the target's state predicted in the previous frame. However, this state cannot be updated correctly after the target disappearance. Therefore, short-term trackers cannot achieve satisfactory performance in long-term tracking. To improve long-term tracking performance, long-term algorithms usually combine a short-term tracker with some re-

- *Haojie Zhao, Bin Yan, Dong Wang and Huchuan Lu are with the School of Information and Communication Engineering, Dalian University of Technology, Dalian, Ganjingzi, 116023, China. E-mail: haojie_zhao@mail.dlut.edu.cn, yan_bin@mail.dlut.edu.cn, wdice@dlut.edu.cn, lhchuan@dlut.edu.cn*
- *Xuesheng Qian is with the CAS Intellicloud Ltd., Shanghai, China. E-mail: xuesheng.qian@intellicloud.ai*
- *Xiaoyun Yang is with the Remark Holdings, London, UK. E-mail: xyang@remarkholdings.com*
- *Corresponding author: Dong Wang, wdice@dlut.edu.cn*

detection components. Although most trackers (e.g., [15], [26], [28]) can search the target over the entire image regardless of the incorrect position state, they are still plagued by incorrect size state, and some of them require a difficult multi-scale search strategy.

Motivated by the discussion above, we propose an effective re-detection method for long-term tracking. This method can estimate the target's state and supply a reference box for the short-term base tracker. Moreover, this method can be combined with most short-term trackers to perform long-term tracking. Experimental results show that most trackers can be improved effectively by combining the proposed re-detection method. We also present a novel verifier to boost our base tracker. This verifier bases on a feature embedding network, improved by the proposed update scheme. The results of ablation studies on this verifier demonstrate its effectiveness.

Our main contributions can be summarized as follows.

- *A novel local-global search framework based on deep networks is proposed to address the long-term tracking task. Both local and global search modules are offline trained and directly used during the tracking process. Our framework is simple yet effective, which could serve as a new baseline for long-term tracking.*
- *An effective re-detection module is proposed for long-term tracking. This module is offline trained and can be combined with short-term base trackers flexibly. We conduct a series of experiments to show that long-term tracking performance can be improved by using our re-detection module.*
- *A novel long short-term updated verifier is developed to boost local tracking. The verifier can be improved obviously by introducing two additional templates and a long short-term update strategy.*
- *Experimental results show that our tracker achieves satisfactory performance on several popular datasets in comparison with state-of-the-art trackers. Extensive ablation studies demonstrate the effectiveness and efficiency of the proposed modules for long-term object tracking.*

This paper builds upon our conference paper [15] and significantly extends it in various aspects. First, we design a new re-detection module to further improve global search performance. Experimental results on several different types of long-term tracker validate the effectiveness of the proposed re-detection module. Second, we improve the local search process by combining an online update method with a feature-embedding verifier and analyze the importance of this novel verifier in local search. Third, we conduct detailed ablation studies on the proposed re-detection module and online updated verifier. Moreover, we have incorporated the large-scale tracking dataset LaSOT, the long-term tracking dataset TLP, and the short-term VOT datasets for more comprehensive evaluations and added more trackers for comparison.

1.1 Related Work

A typical long-term tracker consists of a tracking component and a re-detection component.

Tracking Component in Long-Term Tracker. In most cases, existing tracking methods are directly used as the tracking component. TLD [18] is a classical long-term tracker, which uses a traditional optical-flow-based matching algorithm as the short-term base tracker. In [23], Zhu *et al.* use a detection scheme to track the target. By comparison, most long-term trackers prefer correlation-filter-based methods. In [19], Ma *et al.* use a KCF tracker [29] to perform local short-term tracking. In [20], CSRDCF [30] is used as the base tracker to locate the target within a local search region. Similarly, MUSTer [21] employs an integrated correlation filter, which is based on KCF and DSST [31], to perform translation estimation and scale estimation. PTAV [25] also adopts a correlation filter method as its base tracker. What differentiates PTAV from the aforementioned works is that a deep-learning-based verifier is used to supervise the base tracker during the tracking process.

Recently, deep learning has shown great potential in visual object tracking, and numerous deep-learning-based trackers, especially siamese-network-based trackers, have achieved satisfactory performance. By equipping SiamFC [8] with a re-detection component, Valmadre *et al.* [14] propose a long-term tracker that achieves much better performance than the original SiamFC on the OxUvA benchmark. DaSiam_LT [24] combines DaSiamRPN [32] with a multi-scale searching scheme for long-term tracking. Some works have also integrated a verifier into a tracking component to supervise siamese trackers. Zhang *et al.* [26] use MDNet [4] as a verifier. In this work, we develop an offline learned metric network to verify the tracking results. Because the target's appearance may change drastically in long-term tracking, we design an online update scheme to extend this offline learned verifier.

Re-detection Component in Long-Term Tracker. Different from CMT [22] and MUSTer [21], which use the traditional keypoint matching scheme to detect the target, some works use correlation filters in the re-detection stage. FuCoLoT [20] designs a correlation-filter-based image-wide detector that can estimate the target location as the peak of the response of an entire image. Similarly, the SiamFC+R [14] method attempts to locate the tracked object within a random search region by finding the maximum of the score map.

Most long-term trackers prefer to adopt region-proposal-based methods. The sliding window scheme is a simple and widely used re-detection method. Some region proposals can be generated by setting a suitable stride. In general, the difference of this method used in different trackers is the type of classifier. For example, TLD uses an ensemble of weak classifiers, whereas LCT [19] uses a random ferns classifier. Although the multi-scale sliding window can be used, it is still a coarse re-detection method. Some works have adopted highly complicated schemes to generate region proposals. In [23], Zhu *et al.* use EdgeBox to generate a set of candidate boxes and verify these candidates by using a structured SVM [33] classifier. MBMD uses a sliding window scheme for re-detection. Then, the SiamRPN tracker is used in these windows to further generate candidate boxes.

A similar but not identical method is used in our tracker. We propose a skimming module to accelerate this sliding search process. It initially selects several possible search regions from hundreds of sliding windows with a skimming module. Then, it uses a powerful siamese tracker to re-detect the target within these search regions. However, this re-detection method is still based on the sliding window scheme. Its speed and accuracy depend on the setting of the sliding sampling strategy. For this reason, we also propose a novel offline trained re-detection module. At the re-detection stage, this module directly estimates the target position and the target size over the entire image. We conduct a series of experiments to demonstrate their effectiveness and efficiency.

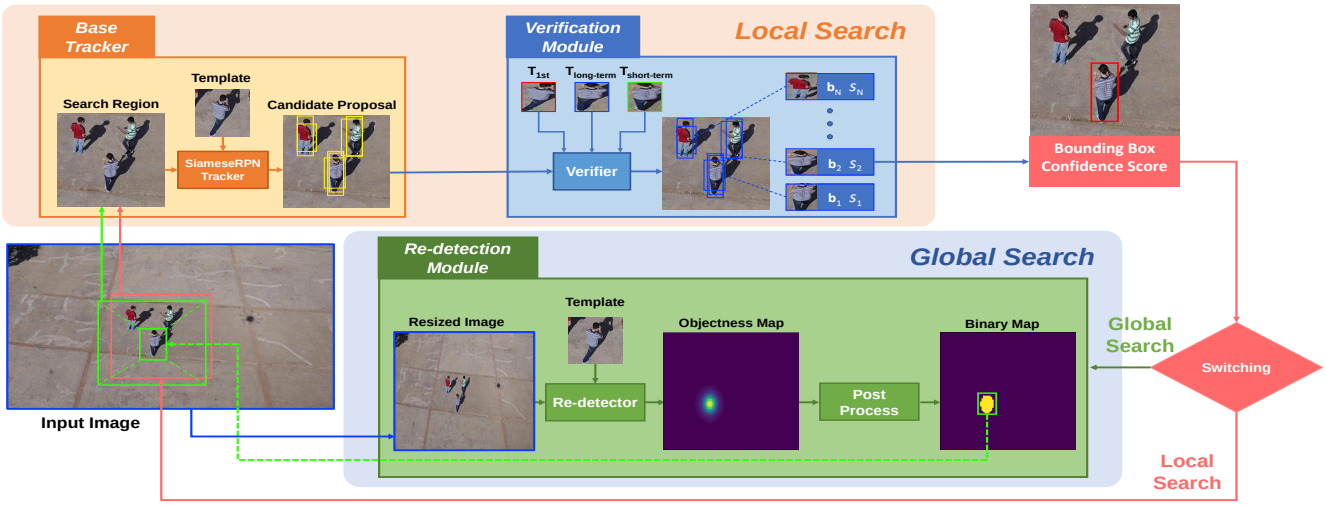


Fig. 1: Overall pipeline of the proposed tracking framework. Better viewed in color with zoom-in.

2 METHODS

In this work, we propose a novel local-global search based tracking algorithm to address the long-term tracking task. The tracking algorithm contains two fundamental modules: local search and re-detection modules. The local search module aims to conduct robust box regression and verification in a local search region, whereas the re-detection module focuses on quickly locating the target over the whole image and estimating the target size when the tracker runs in the global search state.

The overall framework is presented in Figure 1. Our tracker first searches the target in a local search region by using the local search module. After obtaining the best candidate in each frame, our tracker treats the tracked object as *present* or *absent* based on its confidence score and then determines the search state (local search or global search) in the next frame. If the confidence score is higher than a pre-defined threshold, the tracker treats the target as *present* and continues to track the target in the local search region centered by the object location. Otherwise, the tracker regards the target as *absent* and conducts global search in the next frame. To be specific, our global search scheme finds a local search region and then deals with this region using the local search module. To speed it up, we develop a novel re-detection module to efficiently select the most likely local region for the base tracker. The detailed descriptions of our tracking algorithm are presented as follows.

2.1 Robust Local Search with Offline-learned Regression and Verification Networks

Our local search module is composed of an offline-learned SiamRPN tracker and a verification module (shown in Figure 1). The former one generates a series of candidate proposals within a local search region, and the latter one verifies them and determines the best candidate.

SiamRPN. The SiamRPN method (e.g. [9], [32], [34]) improves the classical SiamFC method [8] by introducing a region proposal network, which allows the tracker to estimate the bounding box of variable aspect ratio effectively. In this work, we choose a SiamRPN tracker as our base tracker due to its robustness and efficiency. As illustrated in Figure 1 (Base Tracker), given a target template $\mathcal{Z}$ and a local search region $\mathcal{X}$, the SiamRPN tracker generates a set of bounding boxes $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_N]$ with their corresponding similarity scores $\mathbf{s} = [s_1, s_2, ..., s_N]$, where N denotes the total number of candidates.

Simply, we can directly locate the tracked object based on the bounding box with the highest score in the current frame (i.e., obtain the optimal box as $\mathbf{b}_{i^*}, i^* = \arg\max_i \{s_i\}$. However, this manner makes the tracker unstable and easily drift to some distractors. This phenomenon is mainly attributed to the multi-task learning manner in SiamRPN. The joint learning bounding box predictions and classification scores usually generate accurate box proposals but unreliable scores for the tracking task. Thus, we merely exploit SiamRPN to generate candidate proposals and then infer their confidence scores using an additional verifier.

Verification Network. To ensure the efficiency of our tracking framework, we attempt to exploit an offline-trained verifier based on deep feature embedding [35]. In specific, we learn an embedding function $f(.)$ to embed the target template and the candidate proposals into a discriminative Euclidean space. The discriminative ability is ensured by the following triplet loss,

$$\sum_{i=1}^{M} \left[\|f(\mathcal{Y}_i^a) - f(\mathcal{Y}_i^p)\|_2^2 - \|f(\mathcal{Y}_i^a) - f(\mathcal{Y}_i^n)\|_2^2 + \alpha \right]_+, \quad (1)$$

where $\mathcal{Y}_i^a$ denotes the i-th anchor of a specific target, $\mathcal{Y}_i^p$ is a positive sample (i.e., one of other images of the target), and $\mathcal{Y}_i^n$ is a negative sample of any other target or background. α is a margin value (simply set to 0.2 in this work). $\mathcal{T}$ is the set of all possible triplet pairs in the training set and has cardinality M. The construction of training triplets and the hyper-parameters are presented in Section 2.5.

During tracking, we can determine the confidence scores $[s_1, s_2, ..., s_N]$ of the candidate proposals $[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_N]$ using a thresholding cosine similarity metric as

$$s_i = \max\left(\frac{f^\top(\mathcal{Z}) f(\phi(\mathbf{b}_i))}{\|f(\mathcal{Z})\|_2 \|f(\phi(\mathbf{b}_i))\|_2}, 0 \right), \quad (2)$$

where $\mathcal{Z}$ is the target template in the first frame and $\phi(\mathbf{b}_i)$ denotes the image cropped within the i-th bounding box $\mathbf{b}_i$.

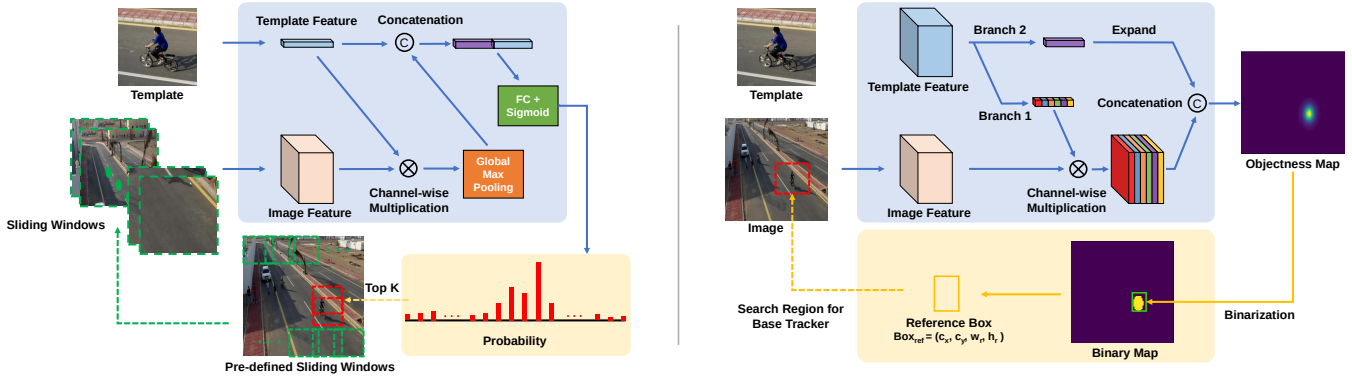


Fig. 2: Frameworks of global re-detection modules: (a) skimming module; and (b) objectness-guided module.

After that, the optimal candidate can be determined as $\mathbf{b}_{i^*}$ ($i^* = \arg\max_i \{s_i\}$), whose corresponding confidence score is s_{i^*}. Finally, we exploit a threshold-based switching strategy to make interactions between local search and global search dynamically. If the score s_{i^*} is larger than a pre-defined threshold θ, our tracker treats the target being *present* and continues to conduct the local search in the next frame. Otherwise, it considers the tracked object as *absent* and then invokes the global search scheme in the next frame.

2.2 Efficient Global Search with Offline-learned Skimming Module

The long-term tracker usually combines a local tracker and a global re-detector and invokes the global re-detector when the object is *absent*. The sidling window technique is widely used to conduct global search [19], [26], by which the local region in each window is utilized to determine whether the object is *present* or not. However, this manner is very time-consuming, especially when deep-learning-based models are used. For example, the recent MBMD tracker [26] (the winner of VOT2018 long-term challenge) merely runs less than 5 fps (very far from real-time performance). To address this issue, we propose a skimming module (shown in Figure 2(a)) to conduct a fast global search by efficiently selecting the most possible candidate regions from a large number of sliding windows.

Given a target template $\mathcal{Z}$ and a search region $\mathcal{X}$, the skimming module aims to learn a function $p = g(\mathcal{Z}, \mathcal{X})$, where p indicates whether the target appears in this region or not. The function $g(.,.)$ is implemented using deep convolutional neural networks (CNN), whose network architecture is presented in Figure 2 (a). Both target template and search region are fed into CNN feature extractors, and then their feature maps are fused and concatenated into a long vector. Finally, the fully connected (FC) layer with the sigmoid function is added to conduct a binary classification. The cross-entropy loss is adopted to train this network.

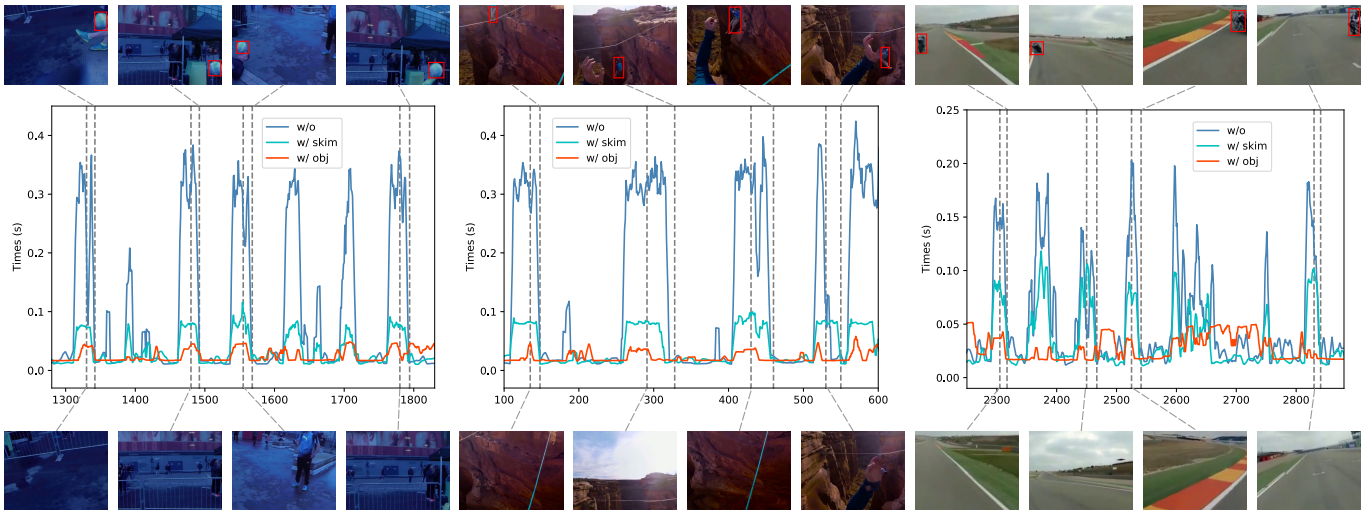
When the tracker runs on the global search, a series of sliding windows are densely sampled. We first apply our skimming module on these regions, and then select the top-K candidates based on their classification margins and discard the remaining ones as distractors. Only selected regions will be further handled using the local search module (Base Tracker+Verifier), which makes our tracker very efficient in image-wide re-detection. In addition, our skimming module could improve the tracker's robustness since it filters out some distractors and alleviates the tracking drift. The parameter K is set as 3 in this work. Figure 3 demonstrates some representative examples, from which we can see that our skimming scheme could significantly reduce the running time when the tracker conducts image-wide re-detection (i.e., the time interval between target disappearance and reappearance).

2.3 Objectness-guided Global Re-detection Module

Although we propose an efficient global search method by designing a novel skimming module, this method is still based on the traditional sliding window strategy. The settings of the sliding sampling strategy, such as stride and window size, can always influence re-detection speed and accuracy. For this reason, we design a novel objectness-guided global search module. This module can directly predict the target size and position over the entire image. Objectness quantifies how likely an image window contains an object of any class [36]. Recent works [37], [38], [39] have proved that objectness can be learned with a convolutional network and can be used for reducing the search space of objects. In this study, we use a fully convolutional network to predict an objectness map and adopt a siamese feature fusion technique to guide the specific object searching process. The flowchart of this re-detection module is illustrated in Figure 2(b).

In our framework, this module is used for coarse location, based on which the base tracker will be applied for accurate location. Our re-detection module is inspired by the detection and segmentation tasks and is specifically designed for the tracking task. We exploit a simple and effective feature fusion structure to make our module target-specific and do not design a heavyweight detection head to predict an accurate bounding box. We directly regression an objectness map based on the target-specific feature map, and then post-process this objectness map to provide a candidate search region for accurate local search. Different from the segmentation and saliency tasks, we do not design our module to predict an accurate mask. We only use several transposed convolution layers to upsample the target-specific feature and predict an objectness map. It is not necessary to predict an accurate mask for determining a candidate search region. Moreover, supervision with bounding box annotation makes label collection and model training much easier than supervision with mask annotation.

Architecture. Instead of inputting a local region patch as many tracking algorithms, we follow the object detection task and feed the entire image to the network. Similar to most siamese-based trackers, a target template is initialized in the first frame, then the target-specific information will be embedded into the feature



(a) freestyle (b) tightrope (c) yamaha

Fig. 3: Effectiveness of the proposed re-detection module. The abbreviations 'w/ skim' and 'w/ obj' denote the trackers with the skimming module and objectness-guided module, respectively. 'w/o' denote the tracker without any re-detection module.

map to conduct one-shot prediction. In our module, we design a feature fusion structure to embed useful information, which uses two complementary parts to embed the template information into the entire image feature on the dimension of channel and spatial. Specifically, the template feature is processed with two additional branches. The first branch is a squeeze-excitation architecture [40]. It squeezes and excites the template feature to produce a set of channel weights. With channel-wise multiplication between these channel weights and the entire image feature, the target-specific information can be recalibrated on the channel dimension. The second branch expands the template feature map, making it be concatenated with the recalibrated image feature. This operation will further embed the target-specific information into each location of the feature map. Finally, the fused feature map is fed to several transposed convolution layers, and a target-specific objectness map can be predicted by this fully convolutional network.

Model Training. This module regresses a resized image X to a Gaussian soft label Y, and it is similar to the deep regression trackers. We thus adapt their training method for our work. The model parameter W is learned by the following square loss

$$L(W) = ||f(X, W) - Y||^2 + \lambda||W||^2. \tag{3}$$

To solve the data imbalance problem of input image, we balance the loss by providing a large weight to positive examples. We also change the response scale in label Y on the basis of the target size to estimate the target size with the predicted objectness map.

Objectness-guided Re-detection. The role of the proposed objectness-guided re-detection is to estimate a reference box for the base tracker, based on which a local search region will be cropped for accurate object localization. As shown in Figure 2(b), we binarize the objectness map with a pre-defined threshold to estimate a reference box (c_x, c_y, w_r, h_r), where (c_x, c_y) is the center coordinate, w_r is the width and h_r is the height. To be specific, we adopt a 'conservative' scale estimation scheme to obtain w_r and h_r, which takes an average between the binary area size and initial box size (groundtruth of the first frame).

Based on the estimated reference box, we apply the cropping scheme proposed in [34] to obtain the search region (c_x, c_y, sz, sz), where

$$sz = 2 \times \sqrt{\left(w_r + \frac{w_r + h_r}{2}\right) \times \left(h_r + \frac{w_r + h_r}{2}\right)}. \tag{4}$$

After cropping the search region, we resize the cropped patch to 255×255 as the input of our tracker.

With the help of the reference box, the performance of the base tracker and the detection result can be improved. In addition, this module can be integrated into many tracking algorithms flexibly and can improve the performance of long-term tracking. We conduct a series of experiments to study the efficiency and effectiveness of the two re-detection methods proposed in this paper. Detailed experimental results are presented in Section 3.5.

2.4 Long Short-term Updated Verifier

The intuitive idea in our local search module is to use a verifier to supervise the tracking process. We adopt a siamese-network-based feature embedding technique (Eq. 1) to verify the tracking results. Although this verification method has shown its effectiveness, it still has a significant drawback. In long-term videos, the appearance of the target may change drastically, especially after tracking failure. However, the original verification module only utilizes a non-update template during tracking. The non-update template will cause the degradation of verification accuracy along with the increase of tracking time. To address this issue, we equip the verification network with a template updating strategy to obtain an improved verifier, namely the long short-term updated verifier.

In [20], several correlation filters updated at different temporal scales are maintained for long-term tracking. On this basis, we maintain three templates updated at different times to boost an existing verifier. A base template T_{1st} is initialized in the first frame, which is the same as that in our original verifier and will never be updated during the entire period of tracking. Immediately, two templates are initialized in the second frame and are updated during tracking, namely, long-term template T_l and short-term template T_s.

Algorithm 1 Long Short-term Update Scheme

Input:
i: frame index, I_l and I_s: update interval, f_l and f_s: flags for updating, d_c and d_s: double check distances, S_v: verification score, θ: pre-defined threshold.
Output:
T_l: long-term template, T_s: short-term template.

if i mod $I_s = 0$ **then**
  $f_s \leftarrow true$;
end if
if i mod $I_l = 0$ **then**
  $f_l \leftarrow true$;
end if
if f_s **and** $S_v < \theta$ **and** $d_c < d_s$ **then**
  if f_l **then**
    $T_l \leftarrow T_s$;
    $f_l \leftarrow false$;
  end if
  $T_s \leftarrow T_{new}$;
  $f_s \leftarrow false$;
end if

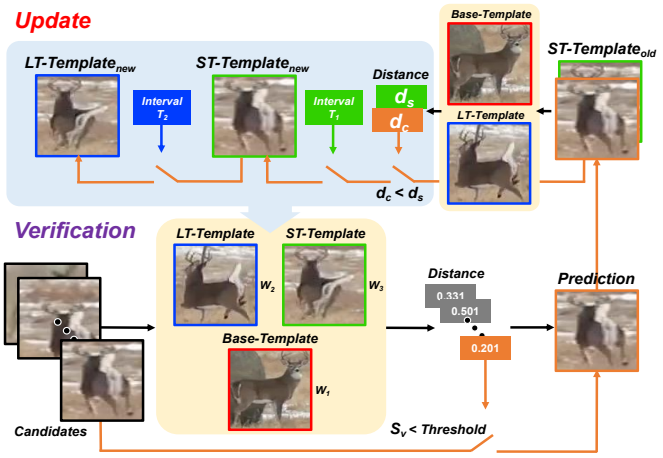


Fig. 4: Designed long short-term updated verifier in our tracker.

Template Update. We set an update interval I_l for the long-term template and an interval I_s for the short-term template. When the time from the last update to the present exceeds any interval, the corresponding template will be allowed for the update. Similar to [15], we adopt a threshold-based verification strategy. A candidate patch can be used for the update if its verification score S_v is less than a pre-defined threshold θ. We adopt a double-check scheme to prevent template contamination, which may seriously damage the performance of our proposed method. This candidate patch must be compared with the current short-term template again. The average distance to the base and long-term templates is calculated for this comparison. If the distance d_c of this candidate patch is smaller than the distance d_s of the current short-term template, then this candidate patch can be used as the new template. The old short-term template is used as the new long-term template, which can ensure the diversity of templates. This update scheme is described by Algorithm 1, where T_{new} denotes the new template.

Verification. The distances $\left[d^i_{1st}, d^i_l, d^i_s\right]$ between a candidate b_i and three templates $[T_{1st}, T_l, T_s]$ are calculated, respectively. The distance is defined as

$$d^i = ||g(b_i) - g(T)||^2, \tag{5}$$

where $g(\cdot)$ denotes an embedding function. Then, the weighted sum of three corresponding distances is computed as the final verification score S^i_v.

$$S^i_v = w_1 \cdot d^i_{1st} + w_2 \cdot d^i_l + w_3 \cdot d^i_s, \tag{6}$$

where w denotes the corresponding weight. Finally, the optimal one is selected from a set of candidates $[b_1, b_2, ..., b_n]$ by finding the minimum verification score. The proposed verifier is presented in Figure 4.

Our long-term tracking algorithm is composed of a base tracker, a verifier and a re-detection module. The base tracker detects the target within a local region. Once the target disappears, the re-detection module will be activated to perform image-wide detection. Instead of using a sliding window strategy to generate single-scale search regions, we propose an objectness-guided re-detection module to predict a possible search region for the base tracker. We also propose a novel long short-term updated verifier that can boost the performance of local tracking. The detailed descriptions of the proposed methods are presented as follows.

2.5 Implementation Details

Network Architectures and Training Datasets. In our work, we adopt SiamRPN++ [34] as our base tracker. MobileNetV2 [41] is used as its backbone network. We also use ResNet18 [42] for the verification and re-detection modules. For the verification module, we downsample the spatial resolution of the feature maps from block3 and block4 to 1×1 by using fully-connected layers, and then concatenate them for the final verification. The resolution of candidate patches is resized to 128×128. For the re-detection module, we use ResNet18 as the backbone network to extract the feature map. We also use two additional branches to further process the template feature map. Both branches are constructed of a global average pooling layer and a 1×1 *conv* layer. We reduce the channel dimension of the fused feature map from 1024 to 256 with a 1×1 *conv* layer and a 3×3 *conv* layer. Then, we use three transposed convolution (4×4, *strides* 2) layers to predict the objectness map. The resolution of the template image is resized to 127×127, and the resolution of the search image is resized to 512×512. The base tracker is trained with large-scale training data, including the training sets of COCO [43], ImageNet DET [44], ImageNet VID, and YouTube-BoundingBoxes [45]. For the verification and the re-detection modules, we use the training set of LaSOT [46] as our training data.

Training Data Preparation. For the regression module, the training data preparation is consistent with Reference [34].

For verification, inspired by the face recognition task, we exploit an offline-trained verifier based on deep feature embedding [35]. Different from face recognition, we construct our training data by using the training set of the LaSOT tracking dataset [46], which contains more than 1100 videos and provides 70 object classes. Benefiting from the various categories, large-scale training samples, and hard negative mining, our verification network can learn useful and discriminative features. We visualize some feature maps learned by our verification network in Figure 6. These feature maps are from the mid-layer of our model and will be processed for final verification by using a global

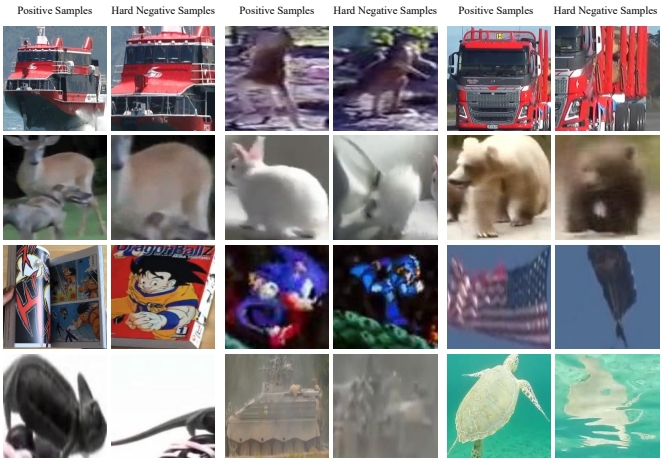


Fig. 5: Positive samples and hard negative samples.

average pooling layer to generate a feature vector. The ablation experiments in Table 6 demonstrate that the verification module significantly contributes to the final tracking performance. In addition to common sampling, we need to choose hard examples as triplet pairs to speed up convergence and boost the discriminative power. To achieve this goal, we first adopt the following sampling strategies for training: (1) randomly choosing one video from the training set and picking its initial target patch as the anchor; (2) stochastically choosing one frame within this video as the positive one; and (3) randomly choosing another frame from the video belonging to a different object class as the negative one. Then, we mine the hard samples to further fine-tune the verification model by the following steps: (1) applying the base tracker and trained verifier to the training set; (2) collecting the false-verified samples as hard samples; and (3) organizing these hard samples for fine-tuning, according to the aforementioned strategies. Some positive and hard negative samples are presented in Figure 5.

For the re-detection module, we adopt the following sampling strategies: (1) randomly choosing one video from the training set and picking two different frames within this video; (2) cropping template image and search image around the target and resizing them to 127×127 and 512×512. This manner is similar to the SiamFC method [8]; (3) generating Gaussian soft label according to the target location and size in the search image.

Optimization. The base tracker, verification module, and re-detection module are trained independently. The base tracker is trained with a stochastic gradient descent optimizer, and the detailed training strategy is consistent with Reference [34]. Both verification and re-detection modules are trained by the Adam [47] optimizer. The batch size for these two modules is chosen as 64. The verification and re-detection modules are trained for 125 and 450 epochs, respectively, with a learning rate of $1e^{-4}$.

3 Experiments

In this work, our tracker is implemented using Python with the PyTorch [48] deep learning libraries. The proposed method is tested on a PC machine with an Intel-i9 CPU (64G RAM) and a NVIDIA RTX2080Ti GPU (11G memory), running in real-time with about **35** frames per second (fps). The tracker equipped with online updated verifier and objectness-guided re-detector is denoted as **Ours**, and the tracker equipped with offline verifier and skimming module is denoted as **SPLT**. Both training and testing codes will be released in the future.

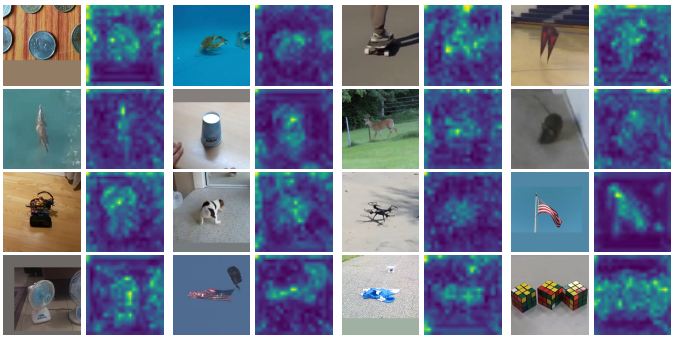


Fig. 6: Feature map visualization of the verification model.

Our tracker is compared with other competing algorithms on several recent long-term and large-scale datasets, including VOT2018-LT, OxUvA, TLP, and LaSOT. The analyses and ablation studies are conducted using the VOT2018-LT dataset. Detailed results are reported as follows.

3.1 Results on the VOT-LT Datasets

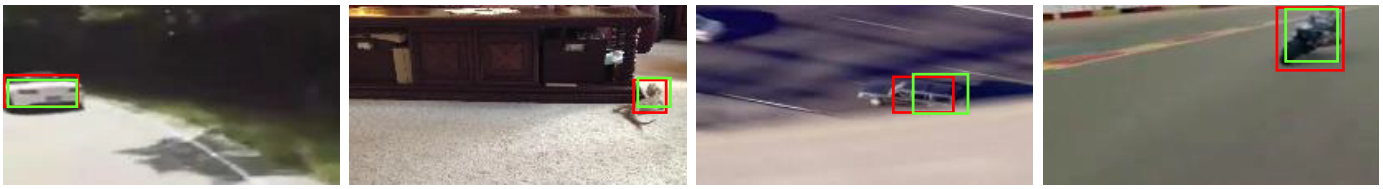
The VOT2018-LT [13] dataset is presented in the VOT 2018 challenge to evaluate the performance of long-term trackers. It includes 35 long-term sequences of various objects with a total frame length of 146847. All sequences consider object disappearance, and the average length of the disappearance interval is 40.6 frames. The target disappearance events in this dataset are more frequent than those in many other long-term datasets. Each sequence of this dataset contains on average 12 target disappearances and is annotated by several visual attributes (e.g., full occlusion, out-of-view, partial occlusion, camera motion, fast motion, scale change, aspect ratio change, viewpoint change, and similar objects). We also use VOT2019-LT for the performance evaluation. The VOT2019-LT [49] dataset is presented by introducing 15 more difficult videos. Its evaluation protocol is the same as that in VOT2018-LT.

In [13], three metrics are proposed to evaluate long-term trackers, namely, *tracking precision* (**TP**), *tracking recall* (**TR**), and *tracking F-score* (**F**). **TP** and **TR** are used to quantify the accuracy of target absence prediction and target re-detection capabilities. Then, **F** is defined based on precision and recall, that is,

$$\mathbf{F}(\tau_\theta) = 2\mathbf{TP}(\tau_\theta)\mathbf{TR}(\tau_\theta)/(\mathbf{TP}(\tau_\theta) + \mathbf{TR}(\tau_\theta)), \quad (7)$$

where τ_θ is a given threshold. $\mathbf{Pr}(\tau_\theta)$, $\mathbf{Re}(\tau_\theta)$, and $\mathbf{F}(\tau_\theta)$ denote the thresholding precision, recall, and F-score, respectively. The F-score plot can be drawn by depicting $\mathbf{F}(\tau_\theta)$ values for all thresholds τ_θ, visually reflecting the long-term tracking performance. On this basis, the primary F-score is defined as the highest F-score at an optimal threshold and is then adopted for ranking different trackers.

Comparison with state-of-the-art algorithms. Table 1 summarizes the comparison results of different tracking algorithms, including SiamRPN++ [34], GlobalTrack [50], MBMD [26], LTMU [51], SiamDW [52], and many VOT trackers [24], [49], from which we can see that our tracker achieves competitive performance in terms of F-score, precision, and recall criteria on VOT2018-LT. The comparison results on VOT2019-LT, shown



(a) following (b) dragon (c) carchase (c) yamaha

Fig. 7: The green box is the reference box generated by the proposed re-detection module. The red box is the annotated groundtruth.

TABLE 1: Comparison of our tracker and competing algorithms on the VOT18-LT [13] datasets. These trackers are ranked from top to bottom by using the F-score measure.

Tracker	F-Score	TP	TR
Ours	0.638	0.669	0.610
SiamRPN++	0.629	0.649	0.610
SPLT	0.616	0.633	0.600
MBMD	0.610	0.634	0.588
DaSiam_LT	0.607	0.627	0.588
GlobalTrack	0.555	0.503	0.528
MMLT	0.546	0.574	0.521
LTSINT	0.536	0.566	0.510
SYT	0.509	0.520	0.499
PTAVplus	0.481	0.595	0.404
FuCoLoT	0.480	0.539	0.432
SiamVGG	0.459	0.552	0.393
SLT	0.456	0.502	0.417
SiamFC	0.433	0.636	0.328

TABLE 2: Comparison of our tracker and competing algorithms on the VOT19-LT [49] datasets.

Tracker	F-Score	Pr	Re	fps
LTMU	0.697	0.721	0.674	13
LT_DSE	0.695	0.715	0.677	13
CLGS	0.674	0.739	0.619	3
Ours-TransT	0.673	0.678	0.667	**27**
SiamDW_LT	0.665	0.697	0.636	3
Ours-R50	0.607	0.611	0.560	**25**
Ours-M	0.590	0.637	0.550	**35**
mbdet	0.567	0.609	0.530	2
SPLT	0.565	0.587	0.544	26
SiamRPNsLT	0.556	0.749	0.443	24
GloablTrack	0.536	0.565	0.510	9
Siamfcos-LT	0.520	0.493	0.549	3
CooSiam	0.508	0.482	0.537	48
ASINT	0.505	0.517	0.494	19
FuCoLoT	0.411	0.507	0.346	7

in Table 2, also demonstrate the competitiveness of our tracker. Our tracker **Ours-M** not only performs better than SPLT, but also achieves a higher F-score than most VOT2019 trackers. We also replace our base tracker with two variants of siamese-based trackers. **Ours-R50** denotes that we replace the MobileNet [41] backbone with Resnet50 [42], and it achieves a 0.607 F-score, which is higher than 0.556 of SiamRPNsLT. SiamRPNsLT adopts the same Resnet50 backbone and SiamRPN++ architecture but uses different long-term components. It suggests that our verification module and re-detection module play important roles in this performance improvement. **Ours-TransT** denotes that we use a modern transformer-based tracker [53] as our base tracker. We also report the tracking speed. It shows that our trackers can achieve competitive performance at a real-time speed, while most other top trackers run slowly. We also conduct a qualitative comparison. The tracking results of the top 5 trackers on VOT2018-LT are shown in Figure 10. The figure shows that our tracking algorithm performs well under various challenging scenarios.

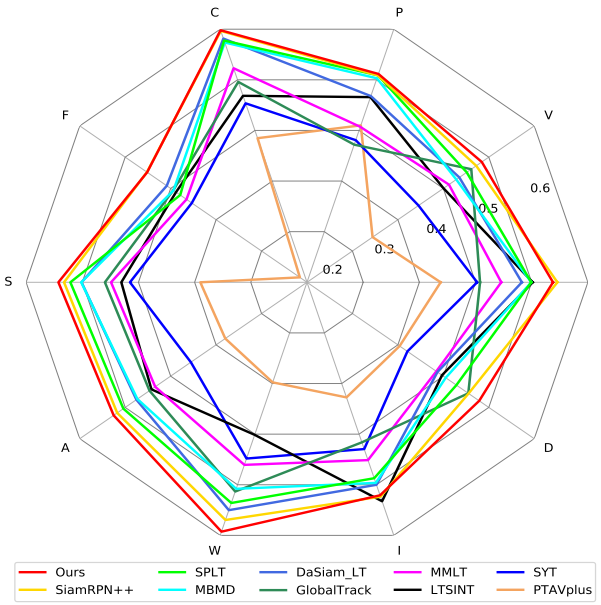


Fig. 8: Quantitative analysis with respect to different attributes. Visual attributes: (O) Full occlusion, (V) Out-of-view, (P) Partial occlusion, (C) Camera motion, (F) Fast motion, (S) Scale change, (A) Aspect ratio change, (W) Viewpoint change, (I) Similar objects, and (D) Deformable object.

Quantitative Analysis on Different Attributes. We conduct this analysis concerning different attributes on VOT2018-LT and report the results of the top 10 trackers in Figure 8. From the figure, our tracker achieves the best performance in most cases, except similar objects. Our tracker significantly outperforms SPLT on the attributes of out-of-view, full occlusion, deformable, viewpoint change, and fast motion.

Tracking Recovery Analysis. We conduct a tracking recovery analysis on VOT2018-LT and report the top 10 trackers in Figure 9. In our experiment, the tracking recovery is defined as the overlap between prediction and groundtruth rises to 0.5 for the first time after target reappearance. Few trackers can recover the tracking process within 50 frames. Only five trackers including ours can recover rapidly within 20 frames.

3.2 Results on the OxUvA Dataset

The OxUvA [14] long-term dataset consists of 366 object tracks in 337 videos, which are carefully selected from the YTBB [45] dataset and sparsely labeled at a frequency of 1 Hz. Compared with the popular short-term tracking dataset (such as OTB2015), this dataset has many long-term videos (each video lasts for an

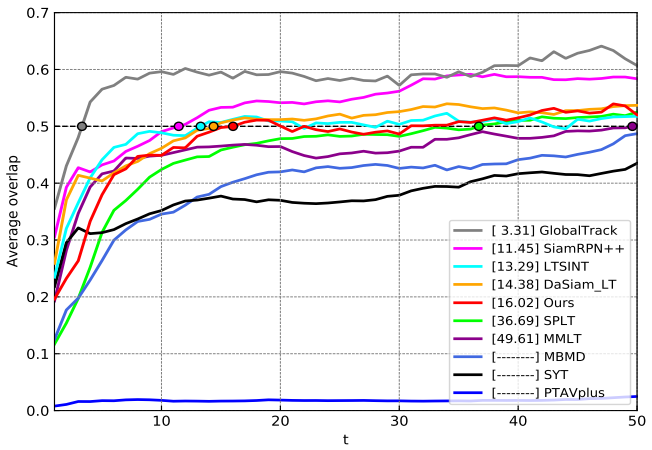


Fig. 9: Average recovery length plots of several competing trackers on the VOT2018-LT dataset.

TABLE 3: Comparison of our trackers and 15 competing algorithms on the OxUvA dataset [14]. The trackers are ranked from top to bottom using the MaxGM measure.

Tracker	MaxGM	TPR	TNR
LTMU	0.751	0.749	0.754
Ours	0.657	0.610	0.707
SPLT	0.622	0.498	0.776
GlobalTrack	0.603	0.574	0.635
MBMD	0.544	0.609	0.485
SiamFC+R	0.454	0.427	0.481
TLD	0.431	0.208	0.895
DaSiam_LT	0.415	0.689	0
LCT	0.396	0.292	0.537
SYT	0.381	0.581	0
LTSINT	0.363	0.526	0
MDNet	0.343	0.472	0
SINT	0.326	0.426	0
ECO-HC	0.314	0.395	0
SiamFC	0.313	0.391	0
EBT	0.283	0.321	0
BACF	0.281	0.316	0
Staple	0.261	0.273	0

average 2.4 minutes) and includes severe out-of-view and full occlusion challenges. In [14], the authors divide the OxUvA long-term dataset into two disjoint subsets, i.e., *dev* (with 200 tracks) and *test* (with 166 tracks) sets. Based on these two subsets, the OxUvA benchmark poses two challenges: constrained and open. For the former one, trackers can be developed merely using the video sequences from the OxUvA *dev* set. For the open challenge, trackers can use any public dataset except for the YTBB *validation* set since the OxUvA dataset is constructed upon it. In [14], three major criteria are considered to evaluate the performance of different trackers, namely, true positive rate (**TPR**), true negative rate (**TNR**), and maximum geometric mean (**MaxGM**). **TPR** gives the fraction of *present* objects that are reported *present* and correctly located, whereas **TNR** measures the fraction of *absent* objects that are reported *absent*. Then, the **MaxGM** rule (8) is defined to synthetically consider both **TPR** and **TNR**, which is adopted for ranking different trackers. For a given tracker, a larger **MaxGM** value means better performance.

$$\mathbf{MaxGM} = \max_{0 \le p \le 1} \sqrt{((1-p)\cdot \mathbf{TPR})((1-p)\cdot \mathbf{TNR} + p)} \ . \quad (8)$$

We compare the proposed tracker with 10 competing algorithms reported in [14], including LCT [19], EBT [23], TLD [18], ECO-HC [7], BACF [54], Staple [55], MDNet [4], SINT [5], SiamFC [8], SiamFC+R [14], and GlobalTrack [**?**]. For a fair comparison, we also include five excellent VOT trackers (i.e., MBMD [26], DaSiam_LT [32], SYT [24], LTSINT [24], and our SPLT). Table 3 shows that our tracker achieves the best performance in terms of **MaxGM** while maintaining competitive **TPR** and **TNR** values. Our tracker performs the best in comparison with other competing algorithms in terms of **MaxGM**, which is the most important metric on OxUvA. Compared with the VOT2018LT winner (MBMD) and its early version SPLT, our method achieves a substantial improvement, with relative gains of 11.3% and 3.5% over **MaxGM**.

Ours *vs* **Short-term Trackers**: We first compare our tracker with some popular short-term trackers. ECO-HC [7], BACF [54], and Staple [55] are three correlation-filter-based short-term trackers with high accuracies and fast speeds. MDNet [4], SINT [5], and SiamFC [8] are three popular deep-learning-based short-term trackers. Compared with these methods, our tracker achieves a very significant improvement in terms of all three quantitative criteria. Table 3 shows that the **TNR** values of the aforementioned short-term trackers are all zeros, which means that these trackers cannot identify the *absent* of the tracked object when it moves out of view or is fully occluded. In principle, these short-term methods cannot meet the requirement of the long-term tracking task. By contrast, our tracker includes an efficient image-wide re-detection scheme and exploits an effective deep-learning appearance model.

Ours *vs* **Traditional Long-term Trackers**: LCT [19], EBT [23], and TLD [18] are three traditional long-term trackers with different hand-crafted features and re-detection schemes. Table 3 indicates that all deep long-term trackers perform better than traditional ones, which means the learned deep features and models are also effective in the long-term tracking task. Our method outperforms the best traditional method (TLD) by a very large margin (0.657 *vs* 0.431 over **MaxGM**).

Ours *vs* **Deep Long-term Trackers**: SiamFC+R [14] equips the original SiamFC tracker with a simple re-detection scheme similar to [56]. Compared with it, our tracker uses an effective re-detection module and exploits a more robust local model to precisely locate the tracked object, and therefore performs better than SiamFC+R. Both MBMD and our methods use SiamRPN-based regressor but different verifiers and different re-detection methods. Our tracker achieves more accuracy and runs much faster than the MBMD method. The tracking performance benefits from the novel re-detection module and online verifier. Our tracker also outperforms three VOT2018LT trackers (DaSiam_LT, LTSINT, SYT) and its early version SPLT by a very large margin.

3.3 Results on the TLP Dataset

The TLP dataset [57] is a large-scale long-term tracking dataset, consisting of 50 high-resolution long videos with an average length of 13529 frames. It was carefully curated with 25 indoor and 25 outdoor videos. Many common scene types are covered in this dataset, such as sky, water, road, and theatre. Moreover, this dataset includes more categories of targets than VOT2018-LT. Precision and success plots are used for evaluating long-term tracking algorithms. Precision plots show the percentage of frames

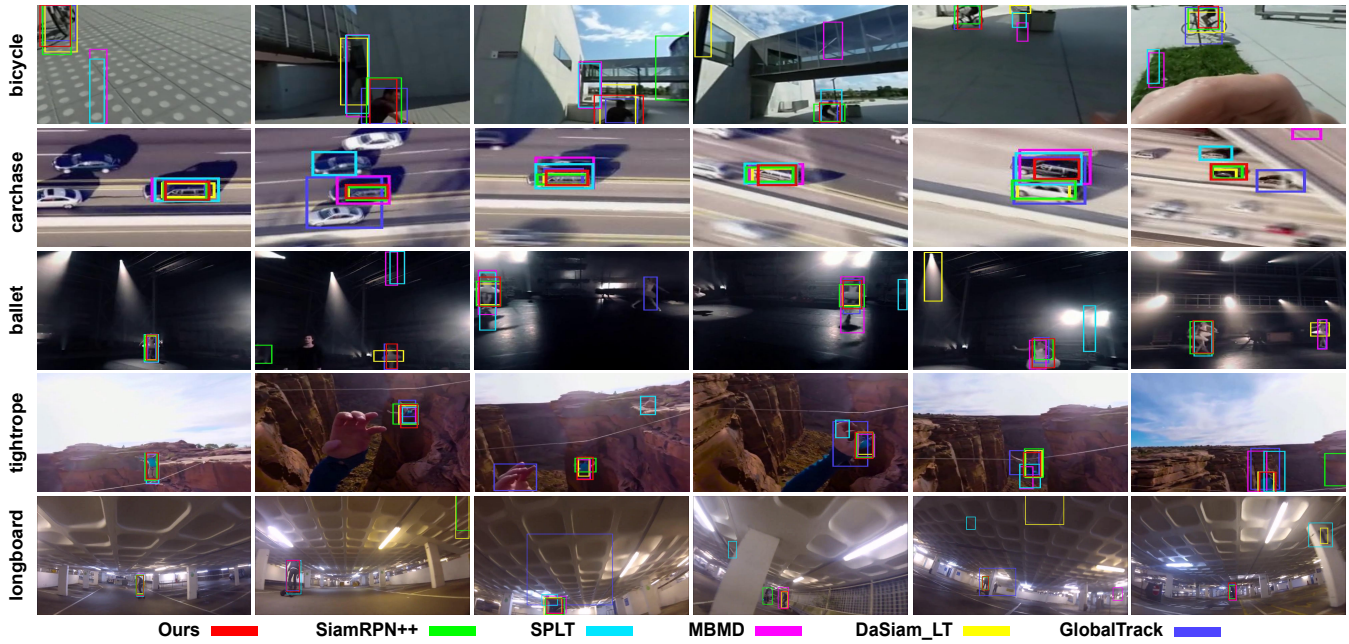


Fig. 10: Qualitative comparison results of several top trackers on VOT2018LT.

TABLE 4: Comparison of our trackers and eight competing algorithms on TLP in terms of success rate (SR, under overlap threshold 0.5), success score, and precision score. The best results are marked in bold fonts. The results of other trackers are reported by [57], [58].

Tracker	SR	Success	Precision
GlobalTrack	0.638	0.520	0.556
Ours	0.598	0.486	0.501
SPLT	0.522	0.416	0.403
SiamRPN	0.515	-	-
MBMD	0.481	-	-
ATOM	0.475	-	-
MDNet	0.423	0.370	0.384
ADNet	0.221	0.223	0.203
ECO	0.219	0.202	0.212
LCT	0.087	0.099	0.072

whose estimated location is within the given threshold distance of the groundtruth. The precision score is computed by using the threshold as 20 pixels. Success plots show the ratio of successful frames as the IoU threshold is varied from 0 to 1. The score is computed as the area under the curve (AUC) of success plots. Success rate measure is also used as the evaluation metric. We evaluate our improved tracker and SPLT on this dataset. Table 4 reports the overall performance of different trackers, showing that our improved tracker performs better than most short-term tracking algorithms and SPLT.

3.4 Results on Short-term Datasets

We first evaluate our tracker on several short-term tracking benchmarks, including VOT-2018 [24] and VOT-2019 [49]. The baseline experiments of VOT-2018 and VOT-2019 follow the auto-reset scheme, however, this scheme is not suitable for long-term trackers equipped with re-detection module. Thus, we have to compare our tracker with other methods using the setting of the unsupervised experiment and adopt the average overlap (AO) as the primary metric. Table 5 reports the comparison results of our tracker with other top-ranked methods in the official VOT reports [24], [49]. Relevant long-term trackers, SPLT [15] and GlobalTrack [50], are also listed for comparison. Then, we also evaluate our tracker using the LaSOT [46] dataset and reports the normalized precision and success plots in Figure 11. LaSOT [46] is a large-scale benchmark and has 280 test videos, which is suitable for evaluating both short-term and long-term tracking tasks. From Table 5 and Figure 11, we can see that our long-term tracker equipped with new modules also achieves competitive performance on short-term benchmarks, especially performing much better than the previous SPLT method.

TABLE 5: Evaluation results on VOT-2018 and VOT-2019.

VOT-18 Trackers	AO	FPS
Ours (Base + Verifier & Re-detector)	0.474	35
Base tracker	0.473	75
GlobalTrack [50]	0.385	6
SPLT [15]	0.383	25.7
SiamRPN [9]	0.472	160
LSART [59]	0.437	1
DRT [60]	0.426	1
ECO [7]	0.402	8

VOT-19 Trackers	AO	FPS
Ours (Base + Verifier & Re-detector)	0.453	35
Base tracker	0.448	75
GlobalTrack [50]	0.375	6
SPLT [15]	0.364	25.7
DiMP [61]	0.508	40
ATOM [62]	0.493	30
SiamMask [63]	0.461	55
SiamDW [52]	0.452	150

3.5 Ablation Study

We conduct ablation analysis to evaluate different components of our tracker using the VOT2018-LT dataset.

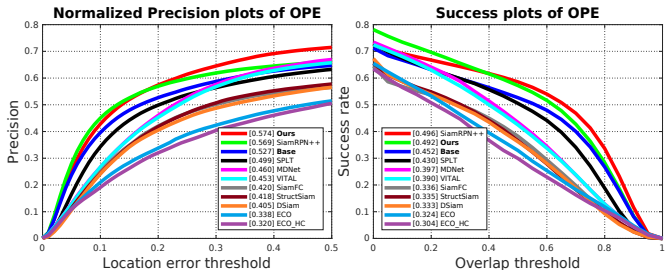


Fig. 11: Evaluation results on LaSOT. 'Ours' denotes our tracker is equated with the verification module and re-detection module. 'Base' denotes that only the base tracker is used for evaluation.

Effectiveness of Different Tracking Components. The proposed tracker follows the long-term tracking framework of *base tracker*, *verifier*, and *re-detector*. It consists of a SiamRPN++ tracker (**T**), a long short-term updated verifier (**V**), and an objectness-guided re-detector (**R**).

To evaluate the contributions of different components, especially the proposed re-detector and verifier, we implement the following variants: (1) **T** denotes our base tracker, which is a SiamRPN++ [34] tracker based on MobileNetV2 [41]; (2) **T+V** denotes the base tracker supervised by the proposed verifier; (3) **T+R** denotes the base tracker equipped with our re-detector and (4) **T+V+R** denotes our final long-term tracker. As shown in Table 6, the proposed verifier and re-detector contribute to the final performance. Moreover, the F-score and tracking recall can be significantly improved by using our re-detector.

TABLE 6: Effectiveness of different components for our tracker.

T	V	R	F	TP	TR
✓			0.500	0.627	0.415
✓	✓		0.515	0.636	0.433
✓		✓	0.605	0.633	0.579
✓	✓	✓	**0.638**	**0.669**	**0.610**

Effectiveness of Long Short-term Updated Verifier. We present a novel long short-term updated verifier that uses three templates updated at different times: **T-1st** denotes the template initialized in the first frame, and this template is never be updated during the entire period of tracking; **T-L** denotes the long-term template, which is updated every other period; and **T-S** denotes the short-term template, which is updated more frequently. The update intervals I_l and I_s are chosen as 1000 and 300, respectively. The weights $[w_1, w_2, w_3]$ of three templates are set to 0.5, 0.3 and 0.2 empirically. Table 7 shows the results of four variants and indicates that the long-term and short-term templates can improve the F-score and tracking recall.

TABLE 7: Effectiveness of different templates in the proposed verifier for our tracker.

T-1st	T-S	T-L	F	TP	TR
✓			0.627	0.667	0.593
✓	✓		0.630	**0.670**	0.595
✓		✓	0.633	0.667	0.603
✓	✓	✓	**0.638**	0.669	**0.610**

As an excellent short-term tracker, RT-MDNet [64] has been used for the long-term tracking task successfully. In [26], RT-MDNet supervises the results of local tracking and re-detection. Thanks to its cleverly designed online update strategy, RT-MDNet can model the appearance of the tracked target well by online parameter fine-tuning. Similarly, the verifier used in this work also benefits from our long short-term update scheme. But unlike RT-MDNet, we only update template images and freeze the parameters of the verification network during the entire period of tracking. In order to study which kind of verifier is more suitable for our framework, we replace the verifier of our tracker with RT-MDNet and re-evaluate this tracker on VOT2018-LT. Table 8 shows the results of different verifiers and indicates the proposed verifier achieves better performance.

TABLE 8: Comparison of our verifier and RT-MDNet on VOT2018-LT [13]. The best results are marked in bold fonts.

Verifier	F	TP	TR	fps
RT-MDNet	0.623	0.659	0.591	22
Ours	**0.638**	**0.669**	**0.610**	**35**

The online network fine-tuning process of RT-MDNet damages the tracking speed obviously. The running speed of the new tracker is only about 22 fps, narrowly satisfying the real-time requirement. Besides, more hyper-parameters are involved in RT-MDNet than in the proposed verifier, such as learning rate, batch size and thresholds of overlap rate. Much time is needed to select a set of appropriate hyper-parameters that may adversely affect the robustness of the tracker.

Effectiveness of Re-detection Methods. In this work, we propose two re-detection methods. The first one is based on the sliding window scheme, accelerated by a novel skimming module. However, this method is still constrained by the setting of the sliding window strategy. Thus, we propose another method to overcome this problem. To analyze their performance, we integrate these two re-detection modules into our tracker and evaluate them on the VOT2018-LT dataset. The comparison results are shown in Table 9 and Figure 12.

In Table 9, we equip our local tracker with two re-detection modules: **+R** denotes the use of the objectness-guided re-detection module, whereas **+skim** denotes the use of the skimming technique (Three windows are selected in our setting). We also equip our local tracker with a crude sliding window method, namely, **+sliding**. The setting of the sliding window strategy in **+sliding** is consistent with that in **+skim**. Comparison results show that the skimming and objectness-guided method can perform better than a crude sliding window method. By equipping the sliding window scheme with our skimming module, the long-term tracking performance can be improved significantly: *tracking F-score*, *tracking precision*, and *tracking recall* have been increased by 9.9%, 12.2%, and 7.9%, respectively. This is because our skimming module can filter out some distractors and alleviate the tracking drift.

TABLE 9: Effectiveness of different re-detection methods. **+sliding** methods come from MBMD [26].

Tracker	F	TP	TR
Ours **+R**	**0.638**	**0.669**	**0.610**
Ours **+skim**	0.612	0.665	0.566
Ours **+sliding**	0.513	0.543	0.487

Considering that the skimming method is constrained by its

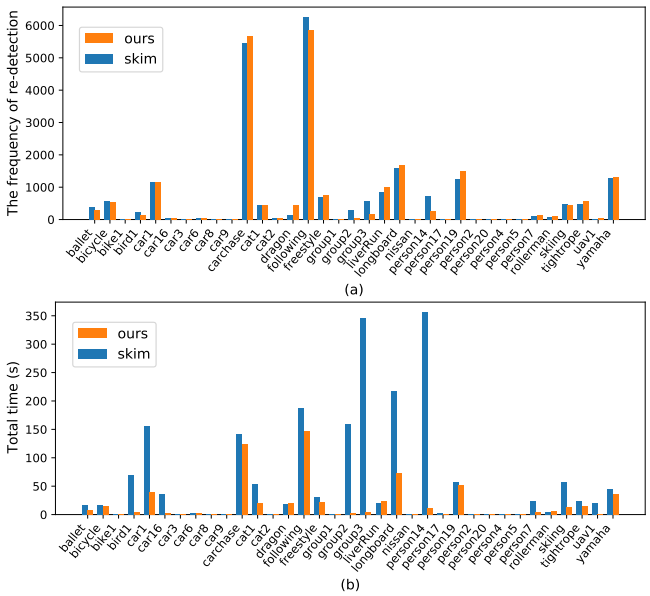


Fig. 12: Comparison results of different re-detection module. (a) Frequency of re-detection in 35 videos of VOT2018-LT. (b) Total time spent on global re-detection in 35 videos of VOT2018-LT.

sliding window strategy, we propose the objectness-guided re-detection method. The tracker equipped with our objectness-guided re-detection module achieves 0.638 *F-score*, and the tracker equipped with our skimming module achieves 0.612 *F-score*. To further analyze their efficiency, we calculate the frequency of re-detection and the total time spent on re-detection, as shown in Figure 12. It shows that the frequency of re-detection is not obviously influenced by re-detection methods. However, the skimming method and the objectness-guided method considerably differ in the total time spent on re-detection. The skimming method always needs more time to find the target during the global search. This phenomenon is particularly obvious in *group2, group3, longboard, and person14*. All these videos have a larger image size, and the target size in these videos is always small. For this reason, the number of sliding windows increases drastically. The statistics suggest that although the skimming module can accelerate re-detection, it is still constrained by the number of sliding windows. Our objectness-guided method is more efficient than the skimming method.

To fully evaluate the objectness-guided re-detector, we integrate this re-detector into several trackers or replace the original re-detection module of some long-term trackers with this re-detector. Then, we re-evaluate the performance of these trackers. The trackers evaluated in this experiment includes a short-term tracker (i.e. SiamMask [63]) and four long-term trackers (i.e., LCT [19], DaSiam_LT [24], SiamVGG [24], and our SPLT). Among these trackers, LCT is a correlation-filter-based tracker, whereas the others are siamese-network-based trackers. As shown in Table 10, all these trackers are improved by using the proposed objectness-guided re-detection method.

Effectiveness of Different Scale Estimation Methods. An important function of our objectness-guided re-detection method is estimating a reference target size for the base tracker. To validate the effectiveness of this function, we conduct the following experiment. We use different reference target sizes in the re-detection stage: **+last** denotes that the reference size is consistent with the last prediction before target disappearance; **+1st** denotes the use of initial box size; **+R** denotes that the reference size is estimated by using our re-detection method. In addition, the target position is estimated by our re-detector in this experiment. Table 11 shows the results.

TABLE 10: Evaluation results of several trackers on the VOT2018-LT dataset. **+R** denotes the tracker equipped with the proposed re-detector.

Tracker	F	TP	TR
LCT+**R**	**0.260**	0.453	0.182
LCT	0.248	0.447	0.171
SiamMask+**R**	**0.546**	0.566	0.528
SiamMask	0.502	0.571	0.447
SPLT+**R**	**0.618**	0.636	0.601
SPLT	0.616	0.633	0.600
DaSiam_LT+**R**	**0.613**	0.646	0.583
DaSiam_LT	0.607	0.627	0.588
SiamVGG+**R**	**0.504**	0.548	0.467
SiamVGG	0.459	0.552	0.392

TABLE 11: Effectiveness of different scale estimation methods and different re-detection methods.

Tracker	F	TP	TR
Ours **+R**	**0.638**	0.669	**0.610**
Ours **+1st**	0.630	**0.671**	0.593
Ours **+last**	0.620	0.659	0.585

Joint Effectiveness of Long-term Components. We also analyze the joint effectiveness and efficiency of our long-term components by equipping several siamese-based trackers with both our verification module and re-detection module. We use two variants of SiamRPN++ [34] and a modern transformer-based tracker [53] for this experiment: **SiamRPN++_M** denotes that the SiamRPN++ tracker uses MobileNet [41] as backbone; **SiamRPN++_R50** denotes that the SiamRPN++ tracker uses ResNet50 [42] as backbone; **TransT** denotes the transformer-based tracker presented in [53]. We conduct this experiment on VOT2019-LT, and the results are shown in Figure 13. The results suggest that the proposed long-term modules can jointly improve the long-term tracking performance, even for a powerful modern tracker, and still ensure a real-time speed. The proposed long-term framework can provide a good trade-off between efficiency and accuracy.

Analysis of computational complexity and model complexity. We also analyse the computational complexity and model complexity of our tracker and compare our tracker with closely related works (i.e., SPLT [15] and MBMD [26]). Both the model size (Params) and floating-point operations (GFLOPs) are reported in Table 12. We briefly explain the notations in Table 12 as follow. (1) In MBMD [26], the base tracker is a modified SiamRPN [9] using a special feature fusion method. The SPLT method, our ICCV version, also follow this base tracker. In this work, we adopt a MobileNet version SiamRPN++ [34] to construct our baseline, which is a more generic choice compared with previous works. (2) The verification module is used to verify the candidates predicted by the base tracker and determine the best one. The MBMD tracker adopts the tracking-by-detection-based MDNet [26] as its verifier. In SPLT, we use the ResNet-50 [42] backbone to

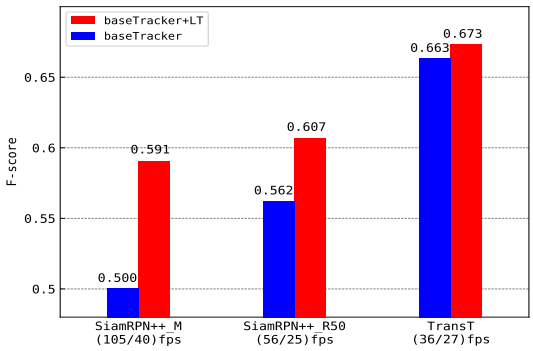


Fig. 13: Joint effectiveness of long-term components on VOT2019-LT. **+LT** denotes equipping with our long-term components.

construct a feature embedder. In this work, we use a similar architecture but adopt ResNet-18 [42] as the backbone for fast inference. (3) For re-detection, MBMD does not exploit any additional module, merely using a simple sliding window strategy to search the target within the entire image. SPLT proposes a novel skimming module and combine it with sliding window for the fast coarse location. In this work, we improve the re-detection method by using an objectness-guided re-detection module to directly conduct the coarse location on the entire image. (4) 'Running' denotes the tracker runs for tracking, and every module will be called dynamically. We run these trackers on VOT-2018LT, and the minimum and maximum of floating-point operations on each frame are shown in Table 12. It is obvious that the max GFLOPs of our tracker is much smaller compared with SPLT and MBMD. What's more, the total model size of our tracker is also smaller than that of the previous SPLT method .

4 Conclusion

This work presents a long-term tracking framework with an effective and efficient local-global search scheme. We design a novel objectness-guided re-detection module for global tracking, which can be combined with other short-term trackers flexibly. This re-detection module can estimate not only the target position but also the target scale, and generate a reference box for the base tracker to determine an appropriate local search region. In the local tracking stage, we use a multi-template and long short-term update scheme to improve a feature-embedding verifier. The proposed long-term tracking algorithm is evaluated on several popular tracking benchmarks: VOT2018-LT, OxUvA, TLP, VOT2018, VOT2019, and large-scale LaSOT. Numerous experimental results demonstrate that our tracker achieves outstanding performance on these benchmarks in comparison with state-of-the-art trackers. We also conduct extensive ablation studies on different tracking components. For the long short-term updated verifier, we analyze the contributions of different templates and compare our verifier to a similar online updated method. For global re-detection, we conduct a series of experiments to analyze our skimming method and objectness-guided method. Experimental results demonstrate the effectiveness and efficiency of the proposed modules for long-term object tracking.

References

[1] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[2] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[3] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3119–3127.

[4] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[5] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1420–1429.

[6] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 472–488.

[7] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6931–6939.

[8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 850–865.

[9] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[10] Y. Wu, J. Lim, and M. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[11] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[12] M. Kristan, A. Leonardis, J. Matas *et al.*, "The visual object tracking VOT2017 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1949–1972.

[13] A. Lukezic, L. C. Zajc, T. Vojír, J. Matas, and M. Kristan, "Now you see me: Evaluating performance in long-term visual tracking," vol. abs/1804.07056, 2018.

[14] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. Smeulders, P. H. Torr, and E. Gavves, "Long-term tracking in the wild: A benchmark," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 692–707.

[15] B. Yan, H. Zhao, D. Wang, H. Lu, and X. Yang, "'Skimming-perusal' tracking: A framework for real-time and robust long-term tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2385–2393.

[16] P. Voigtlaender, J. Luiten, and B. Leibe, "BoLTVOS: Box-level tracking for video object segmentation," *CoRR*, vol. abs/1904.04552, 2019.

[17] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for UAV tracking," in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 445–461.

[18] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[19] C. Ma, X. Yang, C. Zhang, and M. H. Yang, "Long-term correlation tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5388–5396.

[20] A. Lukezic, L. C. Zajc, T. Vojír, J. Matas, and M. Kristan, "FuCoLoT - A fully-correlational long-term tracker," in *Proceedings of Asian Conference on Computer Vision*, vol. 11362, 2018, pp. 595–611.

[21] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "MUlti-Store Tracker (MUSTer): A cognitive psychology inspired approach to object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 749–758.

[22] G. Nebehay and R. Pflugfelder, "Clustering of static-adaptive correspondences for deformable object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2784–2791.

[23] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," in *Proceedings of the IEEE*

TABLE 12: Comparison of computational complexity and model complexity. 'Runtime' denotes the tracker is ran for tracking, and every module will be called dynamically. The dynamic range of floating-point operations per frame is expressed as $min \sim max$.

Tracker	Component	Input Size	GFLOPs	Params
Ours	Base tracker	255×255	7.07	11.15 M
	Verification module	128×128	0.35	11.34 M
	Re-detection module	500×500	5.66	14.09 M
	Runtime	/	$7.42 \sim 21.91$	36.58 M
SPLT	Base tracker	300×300	6.60	13.06 M
	Verification module	128×128	1.80	23.97 M
	Re-detection module	256×256	0.40	3.23 M
	Runtime	/	$8.40 \sim 267.98$	40.26 M
MBMD	Base tracker	300×300	6.60	13.06 M
	Verification module	107×107	0.13	4.43 M
	Runtime	/	$6.85 \sim 6032.73$	17.49 M

Conference on Computer Vision and Pattern Recognition, 2016, pp. 943–951.

[24] M. Kristan, A. Leonardis, J. Matas *et al.*, "The sixth visual object tracking VOT2018 challenge results," in *Proceedings of the European Conference on Computer Vision Workshops*, 2018, pp. 3–53.

[25] H. Fan and H. Ling, "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5487–5495.

[26] Y. Zhang, D. Wang, L. Wang, J. Qi, and H. Lu, "Learning regression and verification networks for long-term visual tracking," *CoRR*, vol. abs/1809.04320, 2018.

[27] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 749–765.

[28] R. Tao, E. Gavves, and A. W. M. Smeulders, "Tracking for half an hour," *CoRR*, vol. abs/1711.10217, 2017.

[29] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[30] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," *International Journal of Computer Vision*, vol. 126, no. 7, pp. 671–688, 2018.

[31] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proceedings of the British Machine Vision Conference*, 2014, pp. 583–596.

[32] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 103–119.

[33] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. Cheng, S. L. Hicks, and P. H. S. Torr, "Struck: Structured output tracking with kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[34] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[35] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[36] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[37] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen, "RON: reverse connection with objectness prior networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5244–5252.

[38] W. Kuo, B. Hariharan, and J. Malik, "DeepBox: Learning objectness with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2479–2487.

[39] T. V. Nguyen, "Salient object detection via objectness proposals," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015, pp. 4286–4287.

[40] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[41] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[43] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[45] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "YouTube-BoundingBoxes: A large high-precision human-annotated data set for object detection in video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7464–7473.

[46] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5374–5383.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations Workshops*, 2015.

[48] A. Paszke, S. Gross, F. Massa *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[49] M. Kristan, A. Berg, L. Zheng *et al.*, "The visual object tracking VOT2019 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 2206–2241.

[50] L. Huang, X. Zhao, and K. Huang, "GlobalTrack: A simple and strong baseline for long-term tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 11 037–11 044.

[51] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang, "High-performance long-term tracking with meta-updater," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6297–6306.

[52] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4591–4600.

[53] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8126–8135.

[54] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1144–1152.

[55] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1401–1409.

[56] J. S. S. III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 322–331.

[57] A. Moudgil and V. Gandhi, "Long-term visual object tracking benchmark," in *Proceedings of the Asian Conference on Computer Vision*, 2018, pp. 629–645.

[58] S. Karthik, A. Moudgil, and V. Gandhi, "Exploring 3R's of long-term tracking: Redetection, recovery and reliability," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2020.
[59] C. Sun, D. Wang, H. Lu, and M. Yang, "Learning spatial-aware regressions for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8962–8970.
[60] C. Sun, D. Wang, H. Lu, and M. Yang, "Correlation tracking via joint discrimination and reliability learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 489–497.
[61] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6181–6190.
[62] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate tracking by overlap maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4660–4669.
[63] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1328–1338.
[64] I. Jung, J. Son, M. Baek, and B. Han, "Real-time MDNet," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 89–104.