Neural Computing and Applications (2023) 35:12611–12625
https://doi.org/10.1007/s00521-023-08422-2

ORIGINAL ARTICLE

RHL-track: visual object tracking based on recurrent historical localization

Feiyu Meng[1] · Xiaomei Gong[1] · Yi Zhang[1]

Received: 1 November 2022 / Accepted: 16 February 2023 / Published online: 4 March 2023

Abstract

Visual object tracking (VOT) is a fundamental and complex problem in computer vision field. In the past few years, the research focus has been shifted from template matching to deep learning models. Especially, the Siamese networks dominate tracking domain in recent years, which take the first frame as the reference and perform object detection and localization in the following frames. However, most of them could not capture target changes due to the lack of strong feature representation abilities. To address these issue, we propose an advanced tracking network in this paper based on recurrent historical localization information. Unlike traditional symmetric structures, we utilize two convolution layers to perform target classification that predicts the initial target center. Then, we apply a gated recurrent unit that fuses multi-resolution features with historical localization information to yield the final optimized target position. Extensive experiments have been conducted on six mainstream datasets: OTB100, GOT-10k, TrackingNet, LaSOT, VOT2018 and NFS, where our tracker exhibits state-of-the-art performances.

Keywords Visual tracking · State estimation · Feature fusion · Recurrent historical localization

1 Introduction

Visual object tracking (VOT) has been spotlighted in recent years with wide industrial applications (e.g., intelligent surveillance [1], autonomous driving [2] and thermal tracking [3, 4], etc.). Given the target location in the first frame with no prior knowledge about the target itself, the tracker aims to locate the same target in subsequent frames using a bounding box. Previously, the correlation filter is the main approach for visual tracking, which distinguishes the target from the background using Fast Fourier Transform (FFT). Later, deep convolution neural networks (DCNNs) begin to dominate and become the indispensable part of the tracking pipeline. Typically, a tracking problem is decomposed into classification and regression tasks, which are implemented by two separate branches. The classification branch generates a scope map to estimate the target's initial position. The regression branch further refines the target position. The two branches are usually deployed in either parallel or serial ways. Although an astonishing advancement has been made around DCNNs (e.g., deeper structure [5], online updating strategies [6], extended data sets [7], etc.), regression of the target state is still the most challenging and critical step due to unpredictable situations, including deformation, occlusion, lighting changes and rotation, etc.

Luckily, the Siamese networks have been put forward to overcome the above difficulties. SiamFC [8] is a cornerstone of Siamese networks by developing the classical two-branch structure with equal weights for target tracking, wherein one branch acquires the template from the first frame, the other branch takes a search patch from the current frame and produces a response map to reflect the degree of relevance between the template and the search patches.SiamRPN [9] organically combined Siamese network with Region Proposal Network (RPN), where the former extracts image features and the latter estimates

Xiaomei Gong have contributed equally to this work.

✉ Yi Zhang
yi.zhang@scu.edu.cn

Feiyu Meng
897173840@qq.com

Xiaomei Gong
2021223040106@stu.scu.edu.cn

[1] Department of Computer Science, Sichuan University, Chengdu, China

target position. Inspired by this, various multi-scale searching strategies have been developed afterward [10–12]. But most of them ignored the fact that regression requires high-level knowledge of the type and pose of the object (the shape of the object may change significantly due to pose changes). And the feature extracted by their shallow networks is unreliable, causing data drifting or target lost under complex situations (refer to Fig. 8 on page 17). Considering this, some online training schemes were presented to update the target changes. Among them, ATOM [13] elucidated a novel tracking architecture with well-designed target estimation and classification components. However, it is trained by a lot of labeled training data and it involves large number of candidate regions with many redundant overlapped areas, which gives rise to high computation cost.

Aiming to design a lightweight and robust tracker with higher estimation accuracy, we propose RHL-Track in this paper. To be specific, our classification branch includes a simple 2-convolutional layers structure with updating strategy. And a gated recurrent unit (GRU) is employed in the estimation branch to optimize the target position. The main contributions of this paper could be summarized as follows:

1. we only calculate and derive the target position with the highest score on the classification branch without computing the traditional back propagation, which ensures the computational efficiency.
2. we devise a feature fusion module (FF), where the shallow and deep features are matched to realize the fusion of spatial and semantic information.
3. we develop a historical localization unit (HLU) to excavate historical target information, so as to improve the tracking accuracy via iterative predictions. Extensive experiments have been conducted on six prevalent tracking benchmarks to demonstrate the effectiveness of our tracker.

2 Related works

In early years, template matching algorithms are traditional means for visual tracking, which calculated similarities between image pairs to estimate target location and handled target variation problem via multi-scale searching schemes. However, they are time-consuming and not flexible. Later, correlation filters began to take their places. An adaptive multi-branch correlation filter tracking method was provided by Li et al. [14] to solve temporal target changes by suppressing the background region. A self-supervised learning-based correlation framework was described by Yuan et al. [15] with a multi-cycle consistency loss. An adaptive spatial-temporal context-aware model [16] was tailored for UAV tracking tasks by reducing the boundary effect. In the meantime, with the growing popularity of deep learning, the Siamese networks begin to prevail and keep setting new records, which attract wide attentions of both researchers and engineers. SiamFC [8] is the pioneering work for Siamese network. SiamRPN [9]improved SiamFC [8] by presenting regional proposal network (RPN). SiamRPN++ [10] further improved SiamRPN [9] by performing depth-wise and layer-wise aggregations. Encouraged by their success, some following works further investigated Siamese networks from different aspects, including deeper backbones [10], more complicated tracking frameworks [17], online updating mechanisms [18, 19], high-speed network [20] and target state estimation oriented methods [21].

More recently, a large number of studies have been published focusing on state estimation, which could be broadly divided into two types: single-stage optimization [22] and multi-stage optimization [23]. Single-stage methods can be further classified into anchor-based regression methods [24] and anchor-free regression methods [25]. The anchor-based regression methods predict one target bounding box within each sliding window with various sizes and aspect ratios (each candidate bounding box corresponds to an offset). A distractor-aware Siamese network with incremental learning was developed by DaSiamRPN [11] to handle long-term tracking. SiamDW [26] leveraged deeper and wider CNNs and devised a new residual module to eliminate the negative impact of padding with controlled receptive field and stride. Albeit effective for some datasets, anchor-based methods need prior knowledge of the anchors, which inevitably leads to higher computation cost. In comparison, anchor-free methods localize the target based on corner or center points. SiamBAN [27] combined classification and regression problems into a unified framework and proposed elliptical fitting. Ocean [28] introduced a feature alignment module to learn an object-aware feature from predicted bounding boxes. SiamCAR [12] is a novel fully convolutional Siamese network with low hyper-parameters. As a typical example of multi-stage optimization method, SPM [21] put forward a coarse-to-fine matching strategy to enhance the robustness and discrimination power. C-RPN [29] declared a cascaded RPNs to fuse features of different levels. High level knowledge was incorporated into the target estimation branch of ATOM [13] through extensive offline learning to measure the prediction accuracy.

The above-mentioned methods push forward the state-of-the-art tracking records. However, the general limitation for Siamese networks is their inability to incorporate background information into the model prediction. Although this problem has been partially solved by using

online training and other template update techniques, they rely on complicated online learning modules which could not be easily implemented in real applications under an end-to-end learning framework. In this background, we analyze the intrinsic attribute of multi-level optimization approach for state estimation in object tracking and present a recurrent optimization scheme to refine the target localization. Unlike symmetric structure, our state estimation branch is trained offline and the classification branch is trained online. Different from some existing methods, which perform multiple matching operations for all candidate regions during reasoning. We estimate the target position from only eight candidate boxes. Our network will be described in Section 3 in details.

3 Method description

Similar to other Siamese networks, our tracker also consists of backbones, the classification branch and the state estimation branch. The classification branch takes the features from both template and search images (that are processed by the backbones) as its input to yield a score map of confidence. To simulate the perturbation in tracking scenario, we add random noise to the ground-truth target location (GT). Then, the state estimation branch outputs eight candidate bounding boxes, and the average position of them is used for the final prediction of the target. As shown in Fig. 1, we use ResNet18 as the backbone for both branches, which is pre-trained on ImageNet for the extraction of target features.

3.1 Target state estimation branch

The main objective of a multi-level optimization network is to minimize the deviation between predicted target box and ground-truth box. To this end, we propose a loop optimization architecture, which derive and update historical target information in a recurrent way. The recurrent state estimation module is shown in Fig. 2 below, which consists of two main components, the feature fusion module (FFM) and the historical localization unit (HLU), in which the HLU performs loop optimization. Information exchange is carried out between the FFM and the HLU. Meanwhile, the deviation of target position is updated iteratively to improve the accuracy of regression. More specifically, given the initial state b^k for the kth iteration, the iterative unit would yield a hidden state h^k in each iteration on the basis of two inputs: v^k (feature fusion results) and h^{k-1}, which will be fed to the head to obtain a predicted deviation of target position $\triangle d$. And the output b^{k+1} of the current iteration is updated as $b^{k+1} = b^k + \triangle d$.

Unlike ATOM, which strove to gain a more accurate target state through maximizing Interaction over Union (IoU) indirectly, we update the target state in an iterative way and output the final optimized state.

3.1.1 Feature fusion module (FFM)

Existing Siamese networks cropped a patch (with background contextual information) from the template image and apply a matching algorithm to search for the most relevant patch in the candidate region via convolution (or channel-wise convolution) operations. The centers of the candidate regions on search image are uniformly distributed. For anchor-free regression algorithms, there are N_{prop} candidate regions. For anchor-based regression methods, there are $k * N_{\text{prop}}$ candidate regions, where k refers to the number of anchor boxes. Previously, the existing methods performed multiple matching operations for all candidate regions in the last stage of reasoning along with additional optimization process to improve the accuracy of regression, which slowed down the tracking speed. Moreover, the incompatible feature matching and bounding box prediction algorithms may further degrade the

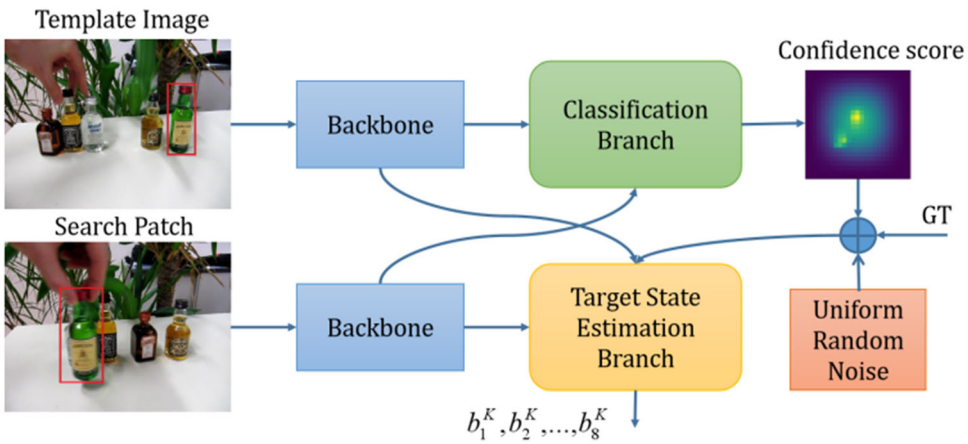


Fig. 1 Overall network architecture

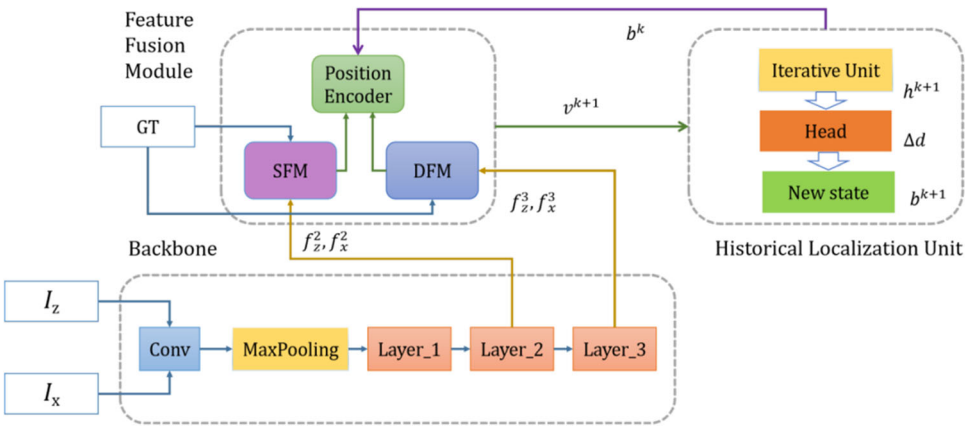


Fig. 2 Flowchart of recurrent state estimation module

tracker's performance. Therefore, a well-designed feature matching method is required to boost the performance of the state estimation module.

The general design principle of a feature fusion module (FFM) should include the following points:

1. It should pay more attention to the selected candidate regions;
2. Suppose the features of the search region and template patch are represented by f and v, respectively. Then, the following equation holds: match$(v, f[C])$ = match$(v, f)[C]$, where C denotes a certain position on the search region, and match( ) stands for a well-designed matching function;
3. The template feature should not include background information, which is regarded as noise;
4. Spatial encoding information should be added into FFM to enhance the spatial information modeling ability.

Guided by the above principles, our feature fusion module (FFM) is comprised of a position encoder (PE), a shallow feature matching module (SFM) and a deep feature matching module (DFM) (as illustrated in Fig. 3 below). As their names imply, SFM and DFM receive the shallow and deep features from the backbone, respectively, to yield more detailed spatial information and constant semantics. Meanwhile, they project the template features and features of searching image (that are extracted by the weight-sharing backbone) to different embedding spaces. Finally, we perform precise ROI pooling operation on the template patch by using only the target information (to obey the third principle).

For position Encoder (PE), to meet the second principle, we perform precise ROI pooling and depth-wise convolution ($\otimes$) on the search feature f and feature vector v as

$$\begin{aligned} &\text{Precision ROI Pooling}(f, C) \\ &\otimes v\langle = \rangle \text{Precision ROI Pooling}(f \otimes v, C). \end{aligned} \tag{1}$$

In the meantime, PE only extracts the feature of the search image within the output area of HLU (to meet the first principle). Besides, the Deviation module (inside PE) attains v_3^{k+1} as $v_3^{k+1} = b^k - b^0$, here b^k is the predicted bounding box after the k^{th} iteration (to meet the fourth principle). Then, v_1, v_2 and v_3 are concatenated into

$$v^{k+1} = concat(v_1^{k+1}, v_2^{k+1}, v_3^{k+1}). \tag{2}$$

Here, v^{k+1} is the final feature fusion results, which will be sent to HLU. We deploy fully connected layers in SFM, DFM and PE to produce global features.

3.1.2 Historical localization unit (HLU)

To realize multi-level optimization, we propose a reusable optimization structure to excavate historical information in a recurrent way. The HLU has an iterative unit and a head. The core block of the iterative unit is a single-stage gated recurrent unit (GRU), which filters historical iteration information. And the head is composed of a fully connected layer, which takes the hidden state from the updated iterative unit as its input, and output the update direction of the target box in current iteration to approach ground-truth annotation.

Our proposed HLU takes the output of FFM to predict a sequence of target states $b^1, b^2, ..., b^K$ based on b^0. For each iteration, a new updated direction $\triangle d$ would be generated to optimize the predicted value as $b^{k+1} = b^k + \triangle d$. The processing flow is as follows:

1. Initialization:
   Initially, we set $v_3^1 = 0$.
2. Input:
   Given the current target state b^k, the fused feature is derived by FFM (as shown in Fig. 3).

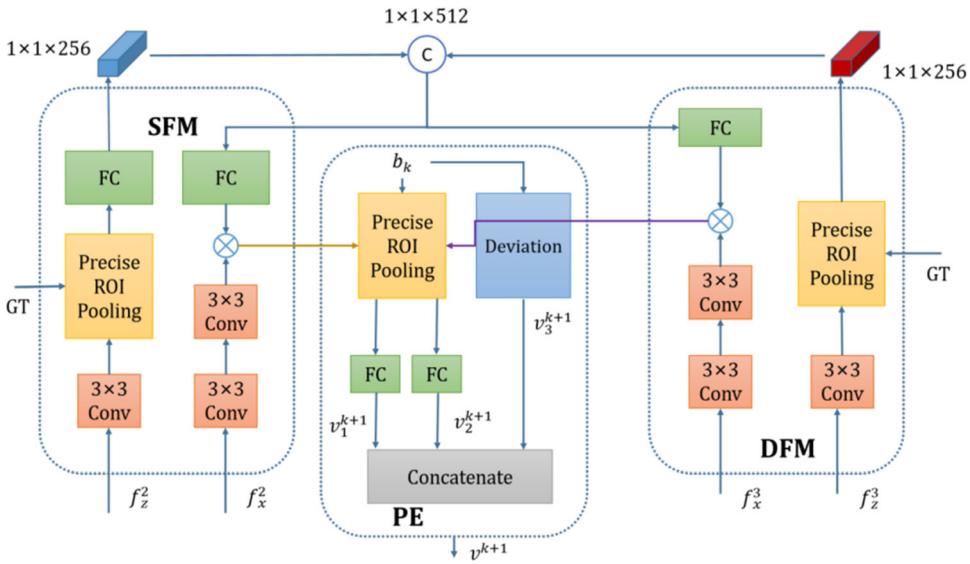


Fig. 3 The structure of feature fusion module

3. State update:

Given the input feature $[v^{k+1}, h^k]$, the GRU in iterative unit updates the target states as follows:

$$\begin{aligned} r^{k+1} &= \sigma(W^r v^(k+1) + U^r h^k) \\ z^{k+1} &= \sigma(W^z v^(k+1) + U^z h^k) \\ \tilde{h}^{k+1} &= tanh(W^h v^(k+1) + U^h(h^k \odot r^{k+1})) \\ h^{k+1} &= (1 - z^{k+1}) \odot \tilde{h}^{k+1} + z^{k+1} \odot \tilde{h}^k. \end{aligned} \tag{3}$$

here, update gate z^{k+1} defines how much of the previous memory to keep around. Reset gate r^{k+1} determines how to combine the new input with the previous memory. h^k is the hidden state after the k^{th} iteration. σ is a Sigmoid function. W^r, W^z, W^h are parameters of the linear model for matrix multiplications. 4, Update the prediction of direction:

The hidden state of the output from the iterative unit predicts the updated direction $\triangle d$, which updates the state as$b^{k+1} = b^k + \triangle d$

3.1.3 Sequence loss function

We assign the prediction sequence b^1, b^2,..., b^K the weights with exponential decay to calculate the error between the result of each iteration and the ground truth. The loss value is defined as

$$\text{loss} = \sum_{k=1}^{K} \alpha^{K-k} * \|b^k - \text{bbox\_}gt\|. \tag{4}$$

Here,bbox_gt is the ground-truth target box on the search image. In our experiment, we set $\alpha = 0.8, K = 5$. α represents the weight of loss of the predicted state obtained in the kth iteration, K denotes the number of iterations [13].

3.2 Target classification branch

Although precise target bounding box could be generated by state estimation branch, it lacks discrimination ability to distinguish the target from the distractions in the background. For this reason, our classification branch aims to boost the discrimination ability. Unlike state estimation branch, the classification branch firstly provides a rough location of the target and then implements online training to adapt to unpredictable target variations over time, which is robust to changing target scales. Considering this, our classification branch takes the features (that is extracted by the third layer of ResNet18) as its input to create useful semantic information. Meanwhile, since we only have the target information in the first frame (sparse sample), thus we only place two convolution blocks in this branch and make the predicted frame as the new sample to estimate the target position in the following frames as follows:

$$s(x, \theta) = \text{conv}_2(\text{conv}_1(x; \theta_1); \theta_2). \tag{5}$$

In (5), x is the feature of the search image, output by the third layer of the backbone; s is the output score map of the target classification branch; θ denotes the network parameter, and conv represents a convolution layer (including the activation function). Then, we locate the position on the search image that corresponds to the point on the score map with the highest confidence score. With the size of the bounding box in previous frame, we could attain the initial value of the target state estimation branch b^0 through the Uniform Random Noise module (as shown in Fig. 1).

3.2.1 Learning objectives

Similar to traditional Siamese networks, in our classification branch, each candidate region on the input frame has a unique score, reflecting its probability of being the target center, which forms the score map s. Since the ground-truth target position is annotated by a bounding box, we thereby model the confidence score using Gaussian distribution. The loss function for online learning is expressed as

$$l = \sum_{k=1}^{m} \alpha_k \sum_{(i,j)\in D_k} (s_{i,j} - y_{i,j})^2 + \sum_{k=1}^{2} \lambda_k \|\theta_k\|^2. \tag{6}$$

Here, D_k represents the score map region of sample k. (i, j) denotes a point on the score map with $s_{i,j}$ as its score. $y_{i,j}$ is the value of the Gaussian distribution obtained at the annotated target location, which is highest at the target center, and decreases from the center to its surrounding areas. M is the total number of samples. α_k refers to the weights for each sample [30]. During update, the parameters of the convolution layers in the classification branch are constrained by the Euclidean norm.

3.2.2 Online updating

The most popular optimization method is stochastic gradient descent (SGD) method, which is a first-order method and has a fast descent speed but slow convergence. Instead, we choose Gauss–Newton (GN) method as our approximation method. Although it is a second-order method, it has fast convergence speed, since we only need to calculate the first derivative instead of the complex Hessian matrix. The advantage of GN over SGD is proved by ablation experiment in the following Sect. 3.6.3. The residual function is defined as follows:

$$r_k(\theta) = \sqrt{\alpha_k}(s_k - y_k), k \in \{1, 2, \ldots, m\}. \tag{7}$$

where α_k is the same weights for the samples in (5). x_k is the kth sample, s_k is the predicted score map of x_k by the network, y_k is the corresponding score label of x_k. Furthermore,

$$\begin{aligned} r_{m+1}(\theta) &= \sqrt{\lambda_1}\theta_1 \\ r_{m+2}(\theta) &= \sqrt{\lambda_2}\theta_2. \end{aligned} \tag{8}$$

Here, λ_1, λ_2 are the regularization weight coefficients corresponding to the parameters of different convolution layer in (6). Then,

$$r(\theta) = \text{concat}(r_1(\theta), r_2(\theta), \ldots, r_m(\theta), r_{m+1}(\theta), r_{m+2}(\theta)). \tag{9}$$

The loss function in (6) could be written as

$$l(\theta) = \|r(\theta)\|^2. \tag{10}$$

We make use of the quadratic Gauss–Newton approximation $\tilde{l}_\omega = l(\omega + \triangle\omega)$, which is calculated by the first-order Taylor expansion of the residuals $r(\omega + \triangle\omega) = r_\omega + J_\omega \triangle \omega$ at the current ω:

$$\tilde{l}(\triangle\omega) = \triangle\omega^T J_\omega^T J_\omega \triangle \omega + 2 \triangle \omega^T J_\omega^T r_\omega + r_\omega r_\omega. \tag{11}$$

here, we define $r_\omega = r(\omega)$ and $J_\omega = \partial r/\partial\omega$ as the Jacobian of r at ω. $\triangle\omega$ is the increment of parameter ω. Equation (11) forms a positive definite quadratic function, which could be solved via conjugate gradient method.

3.2.3 Network architecture

The $conv_1$ in (5) is comprised of 64 1×1×256 convolution blocks without any activation functions. $conv_2$ in (4) has 1 4×4 convolution block, and it utilizes the following nonlinear activation functions:

$$a(t) = \begin{cases} t, t > = 0 \\ 0.05 * (e^{t/0.05} - 1), t < 0 \end{cases} \tag{12}$$

3.3 Implementation details

In this section, we will describe the training and reasoning processes in details.

3.3.1 Training

We adopt ResNet18 as our backbone, which is pre-trained on ImageNet, and the parameters keep unchanged afterward (including the initial values of the hidden states of GRU). The entire network is trained end-to-end using image pairs, especially the state estimation branch is trained offline and the classification branch is trained online. Besides, we add Gaussian noise on the annotated results to simulate the initial rough estimation results so as to yield eight candidate bounding boxes, where the IoU of any of them to the ground-truth box is larger than 0.1. The training objective is to calculate the deviations between the eight boxes and the real target position. We train our model 50 epochs using ADAM optimizer. The initial learning rate is set to 10^{-3}, and the learning rate decays by a factor of 0.2 for every 20 epochs. Similar to existing Siamese tracking networks, we use LaSOT, TrackingNet, GOT-10k and COCO as the training sets.

3.3.2 Reasoning

We perform online training to update classification branch. For the first frame, we carry out translate, rotation,

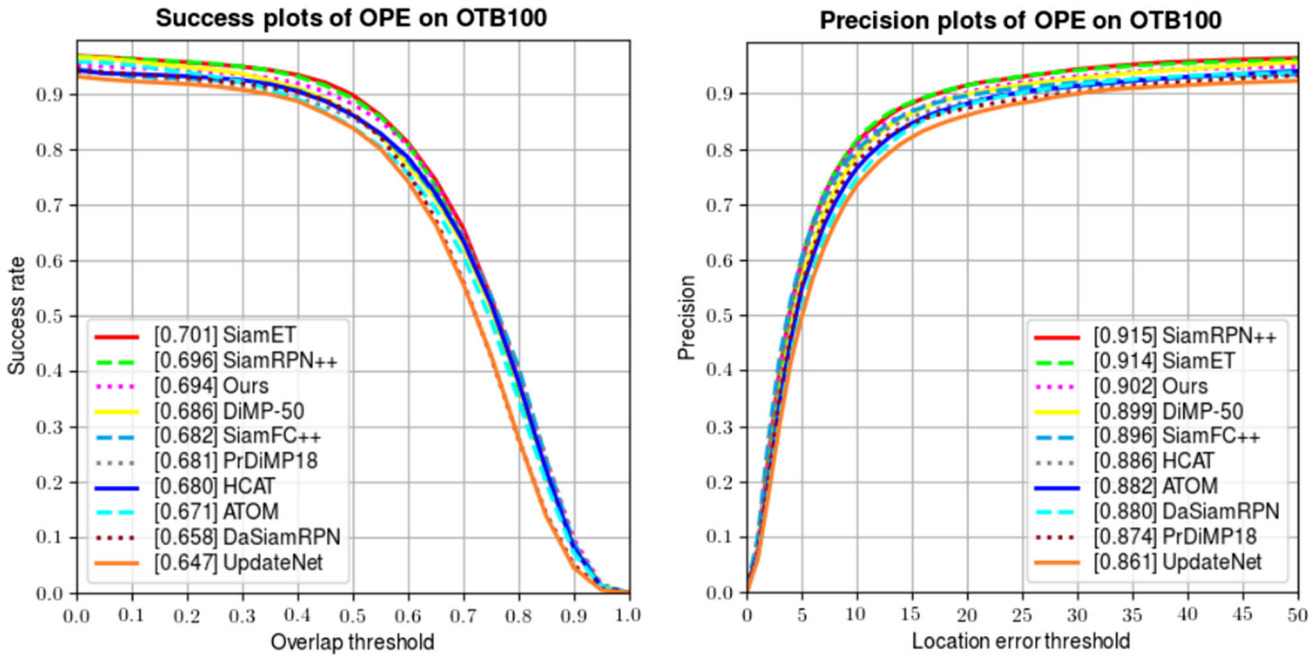


Fig. 4 Comparison of performances on OTB100

blurring, etc., for data augmentation purpose to produce 30 initial training samples and optimize the last convolution layer of our network. We set the first frame as the template. Based on classification branch, the tracker is able to derive the position with the highest confidence score, which is combined with the width and height of the bounding box in the previous frame to generate the initial target box. During reasoning, we perform five iterations of optimization to yield eight candidate boxes. Then, the average position of them is regarded as the final position of the target.

3.4 Experiment and analysis

In this section, we test the performances of our tracker on six benchmarks and compare with other popular trackers, including SiamRPN++ [10], SiamFC++ [31], ATOM [13], DiMP [32], SiamBAN [27], PrDiMP-18 [33],Siam-CAR-Staple [34], SiamET [35] and SiamLAN [20]. We use Ubuntu 20.04, GeForce GTX 2080Ti GPU, Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz as our hardware platform. We achieve a running speed of 32 fps, while ATOM has only 26 fps.

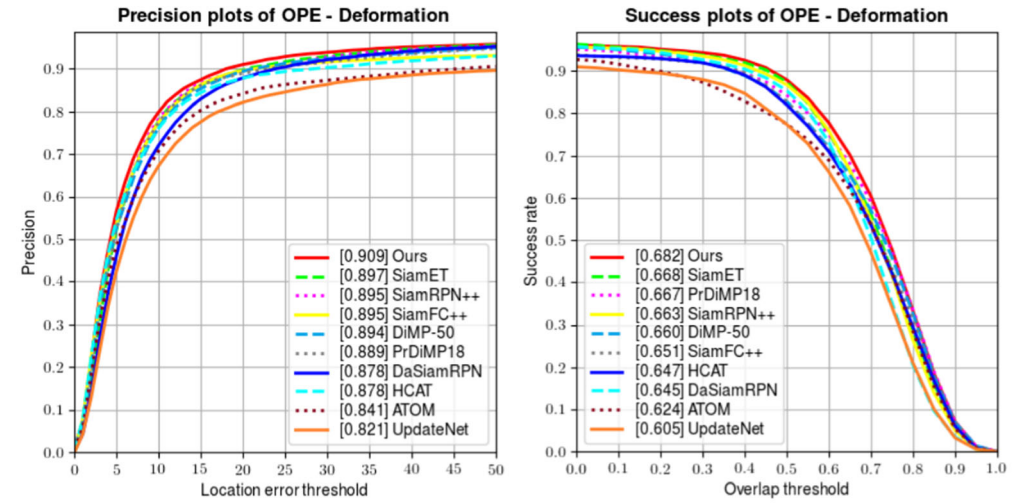
3.4.1 OTB100

OTB100 [36] is a popular tracking dataset, which consists of 100 short-term videos of complex scenes. We compare our results with other state-of-the-art tracker (including SiamFC++, DiMP, SiamRPN++ and newly published SiamET, etc.) in terms of success rate and precision. The experimental results are listed in Table 1 and Figure 4. We rank 3^{rd} in both success and precision.

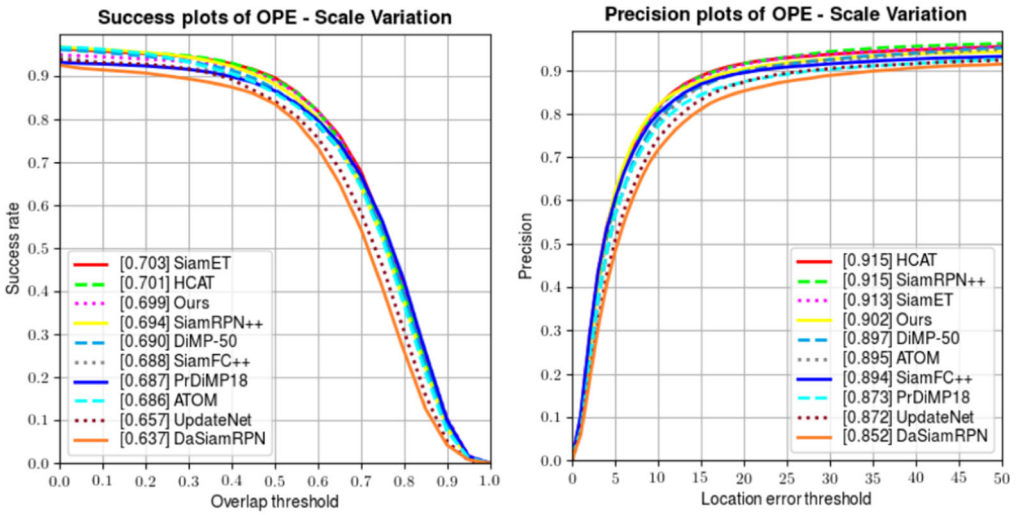
SiamRPN++ utilized a spatial aware sampling strategy and performed layer-wise and depth-wise aggregation to ensure the tracking accuracy. SiamET developed a template enhancement module to keep track of the target variations. Therefore, they achieved better results in short-term tracking. In addition, they all adopt ResNet-50 as their

Table 1 Comparative results on OTB100

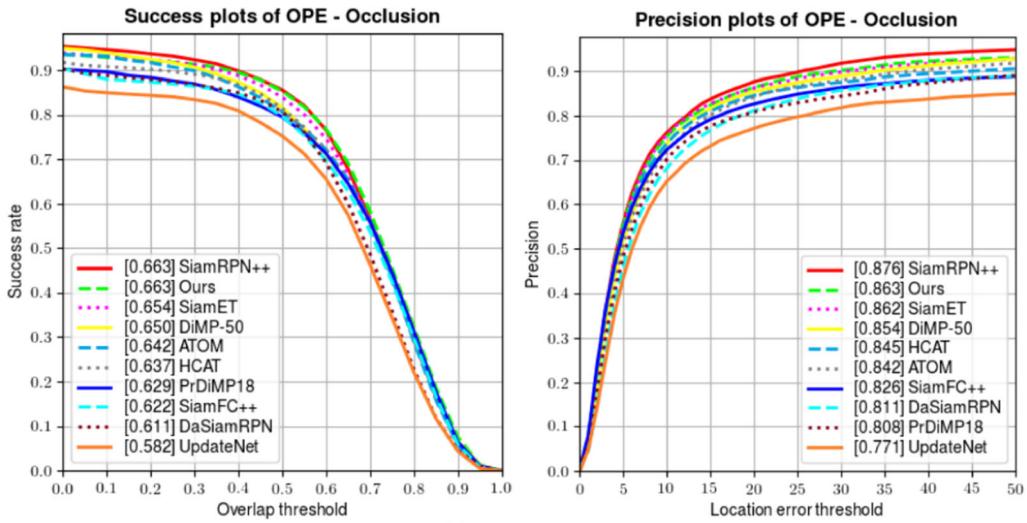
Trackers	SUCC	Precision
GradNet [37]	0.639	0.861
UpdateNet [38]	0.647	0.861
ATOM [13]	0.671	0.882
SiamFC++ [31]	0.682	0.896
GCT [18]	0.639	0.853
DiMP-50 [32]	0.660	0.899
TRAT-18 [30]	0.663	0.873
DaSiamRPN [11]	0.658	0.880
SiamLAN [20]	0.667	0.891
E.T.Track [39]	0.678	–
HCAT [40]	0.681	–
Lighttrack [41]	0.662	–
Zhou [42]	0.677	0.891
Gao [43]	0.667	0.784
SiamRPN++ [10]	0.696	0.915
SiamET [35]	0.701	0.914
Ours	0.694	0.902



(a) Deformation



(b) Scale Variation



(c) Occlusion

Fig. 5 Comparisons of different attributes on OTB100

Table 2 Comparative results on VOT2018

Trackers	Accuracy	Robustness	EAO
SiamFC [8]	0.512	0.670	0.201
DRT [45]	0.355	0.201	0.355
RCO [44]	0.507	0.155	0.376
UPDT [46]	0.536	0.184	0.379
MFT [44]	0.505	0.140	0.385
ATOM	0.590	0.203	0.401
ASC-DCF [47]	0.511	0.155	0.403
SiamSMDFFF [48]	0.532	–	0.398
SiamRPN++ [10]	0.600	0.234	0.414
SiamFC++ [31]	0.587	0.183	0.426
SiamLAN [20]	0.596	0.258	0.384
SiamET [35]	0.596	0.155	0.480
Ours	0.601	0.164	0.427

backbones, while we use ResNet18. Therefore, they achieved slightly better results than us in short-term tracking.

To further validate the effectiveness of our tracker, we compare the performances of different trackers on OTB100 in dealing with various unpredictable challenges (shown in Fig. 5), including (a) Deformation, (c) Scale variation and (c) Occlusion. Compared with other popular methods, we also achieve competitive performances.

3.4.2 VOT2018

VOT2018 [44] (Visual Object Tracking Challenge 2018) [44] includes 60 different kinds of challenging videos. The performance of each tracker is evaluated by two indexes: Accuracy (average overlap over successfully tracked frames) and robustness (failure rate). Additionally, a comprehensive index called EAO is formed by combining the above two indexes. The experimental results are shown in Table 2, our tracker ranks first in accuracy and second in EAO. Besides, a comparison of all attributes on VOT2018 is shown in Fig. 6. The attributes include camera motion, illumination change, occlusion, size change and motion change, which are all annotated per-frame. These results indicate that our tracker is adaptable to complex situations by capturing target changes.

3.4.3 LaSOT

LaSOT [7] is a large-scale dataset including 1,400 videos of long sequences (with 2,512 frames on average) with different unpredictable challenges (e.g., partial occlusions, temporal disappearances, etc.). It has 70 categories, and each of them has 20 individual videos. We evaluate our tracker on its 280 testing videos in terms of success rate, precision plots and normalized precision plots. In this test, DiMP-50 wins the first place in all 3 metrics, while our tracker ranks second. It is worth noting that SiamET and

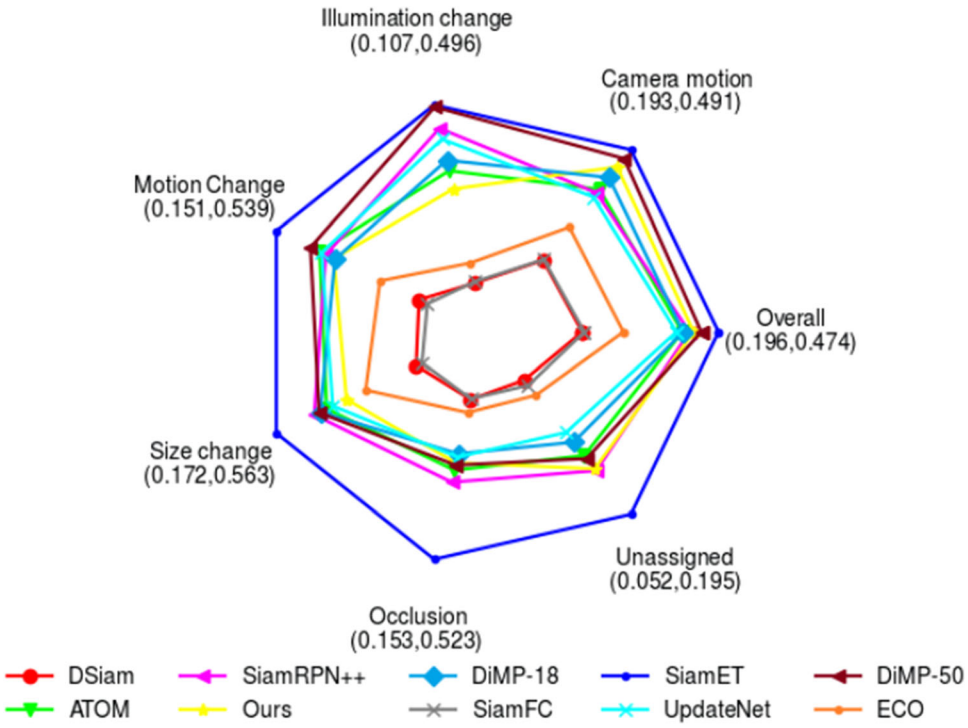
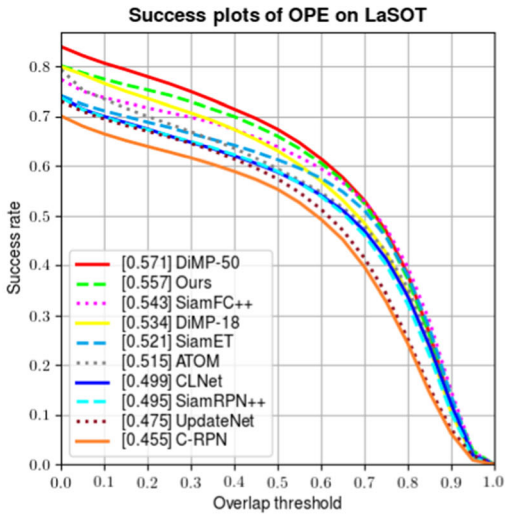
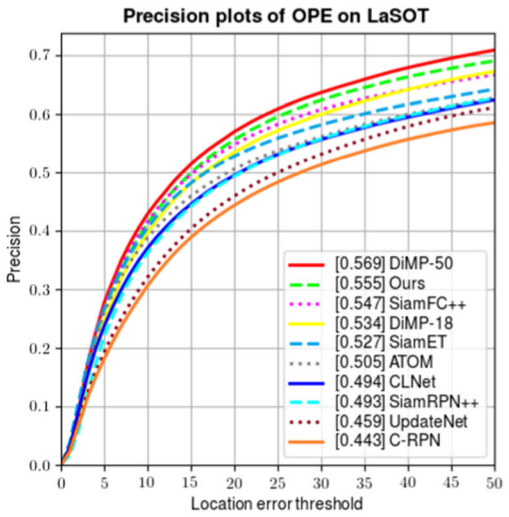


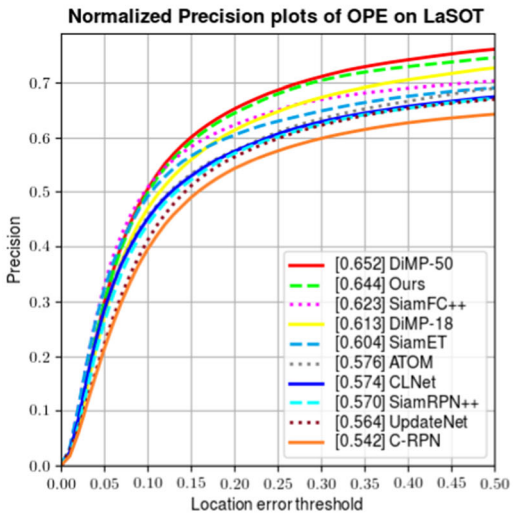
Fig. 6 Comparison of EAO on VOT2018



(a) Success plot on the LaSOT



(b) Precision plot on the LaSOT



(c) Normalized Precision plot on the LaSOT

◀**Fig. 7** Comparisons of different attributes on LaSOT

SiamRPN++ achieved higher results than ours on OTB100 (short videos). However, we outperform them on LaSOT (long videos) by some safe margins. This result reflects the general shortcomings of existing Siamese networks in updating target changes in long time duration. By contrast, our proposed historical localization unit (HLU) captures the target changes in a more precisely and timely manner.

Table 3 Comparative results on LaSOT

Trackers	Succ. (AUC)	Prec	Norm. Prec
ECO [49]	0.324	0.301	0.338
ATOM [13]	0.516	0.506	0.577
MDNet [50]	0.397	0.373	0.461
SiamRPN++ [10]	0.496	0.492	0.570
SiamBAN [27]	0.515	0.522	0.599
UpdateNet [38]	0.476	0.459	0.564
SiamCorners [51]	0.480	–	0.555
DiMP-18 [32]	0.534	0.534	0.613
SiamLAN [20]	0.469	0.467	0.551
Lighttrack [41]	0.555	0.561	–
SiamET [35]	0.521	0.527	0.604
Zhou [42]	0.520	0.530	0.603
Gao [43]	0.469	0.467	0.551
SiamFC++ [31]	0.544	0.547	0.623
DiMP-50 [32]	0.571	0.569	0.652
Ours	0.557	0.555	0.644

Table 4 Comparative results on GOT-10k

Trackers	AO	$SR_{0.5}$	$SR_{0.75}$
SiamFC [8]	0.348	0.353	0.098
ECO [49]	O.316	0.309	0.111
ATOM [13]	0.556	0.634	0.402
MDNet [50]	0.299	0.303	0.099
SiamRPN++ [10]	0.518	0.618	0.325
ROAM [53]	0.436	0.466	0.164
ROAM++ [53]	0.465	0.532	0.236
SiamTPN [54]	0.598	–	–
TRAT [30]	0.608	0.720	0.467
DiMP-18 [32]	0.579	0.672	0.446
SiamFC++ [31]	0.595	0.695	0.479
DiMP-50 [32]	0.611	0.717	0.492
Ours	0.614	0.718	0.504

Table 5 Comparative results on TrackingNet

Trackers	Succ. (AUC)	Prec	Norm. Prec
SiamFC [8]	0.559	0.518	0.652
ECO [49]	0.554	0.492	0.618
ATOM [13]	0.703	0.648	0.771
MDNet [50]	0.606	0.565	0.705
DiMP-18 [32]	0.723	0.666	0.785
UpdateNet [38]	0.677	0.625	0.752
C-RPN [29]	0.699	0.619	0.746
SiamCorners [51]	0.695	0.647	0.763
Lighttrack [41]	0.725	0.695	0.779
SiamRPN++ [10]	0.733	0.694	0.800
DiMP-50 [32]	0.740	0.687	0.801
Ours	0.734	0.683	0.797

Table 6 Comparative results on NFS

Trackers	AUC
MDNet [50]	0.422
C-COT [57]	0.488
UPDT [46]	0.536
ECO [49]	0.466
ATOM [13]	0.584
SiamCorners [51]	0.537
HDT [58]	0.398
FCNT [59]	0.391
DaSiamRPN [11]	0.395
TRAT-18 [30]	0.585
E.T.Track [39]	0.590
Lighttrack [41]	0.553
SiamBAN [27]	0.594
DiMP-18 [32]	0.610
SiamCAR [12]	0.618
DiMP-50 [32]	0.619
Ours	0.623

It's worth noting that SiamET and SiamRPN++ achieved higher results than ours on OTB100. We outperform them on LaSOT by some safe margins. The reason is that we rely on HLU to update the template so as to localize the target more precisely. This results demonstrate the robustness of our tracker in dealing with long-term videos.

As shown in Fig. 8, we select three video sequences (four frames for each) from LaSOT and compare our results with other state-of-the-art methods (including C-RPN [29], ATOM [13], SiamFC++ [31], SiamBAN [27] and UpdateNet [38]). For the video sequences in the first row, initially, all bounding boxes are targeted at the tank (first column). When the second tank approaches the target tank (second column), UpdateNet (blue) and C-RPN (purple) enlarge their initial bounding boxes to embrace them together. Next, when the tanks change their poses (third column), ATOM (light blue) only captures the head of the tank, while SiamFC++ basically loses the target. For the second sequence (in the second row), the rider is blocked by the slope. ATOM (light blue) enlarges its searching scope, trying to enclose the target. When the target shows up again, all other methods lose track of the rider by drifting to other parts of the background, only our method still sticks to it. For the third video (the third row),

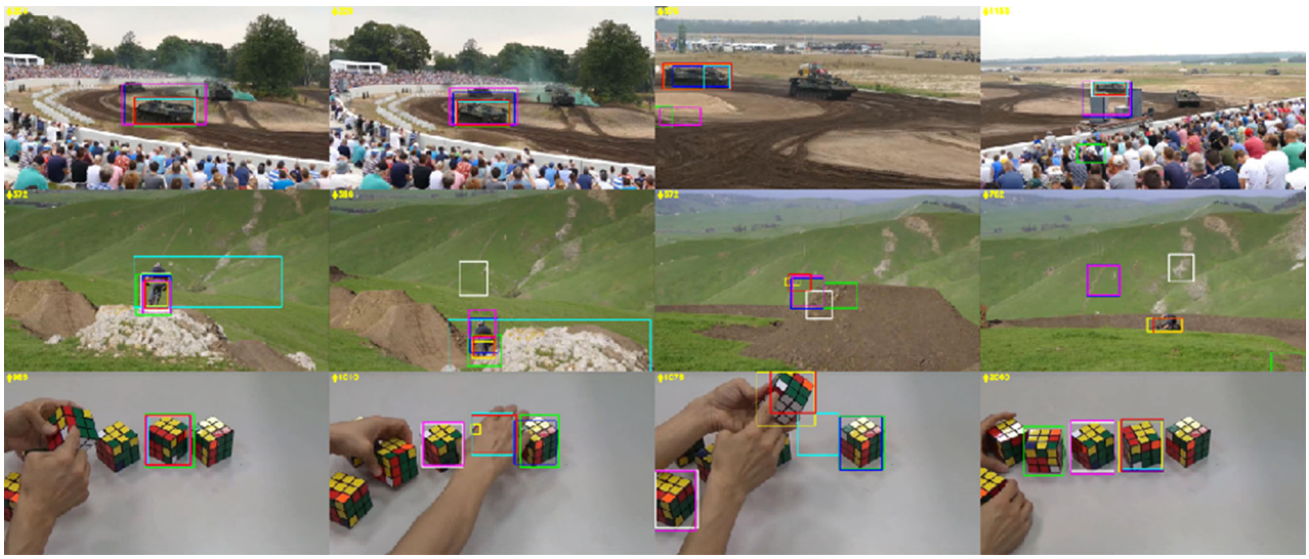


Fig. 8 Qualitative comparison of tracking results on LaSOT among other popular methods. Ground truth: Yellow; SiamBAN: White; SiamFC++: Green; UpdateNet: Blue; ATOM Light blue; C-RPN: Purple; Ours: Red (Color figure online)

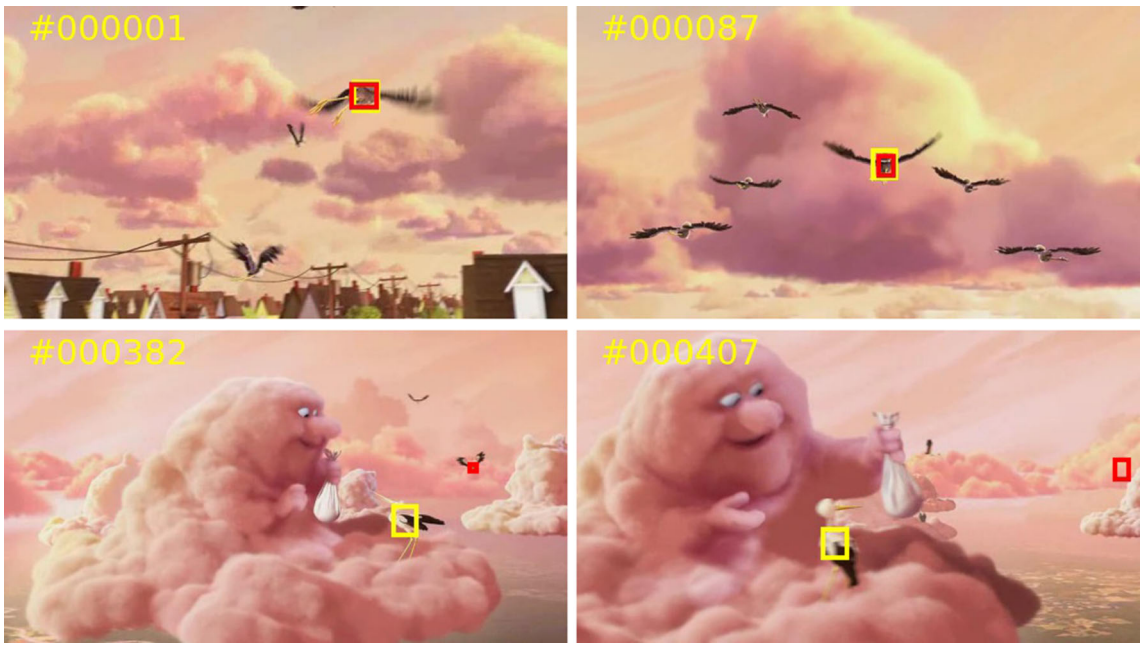


Fig. 9 Illustration of failure case (Color figure online)

when the tracked target is occluded by human hands, other methods are confused by surrounding similar targets and lose the initial matrix. Apparently, compared with other methods, our tracker manifests appealing performance in dealing with interference, partial occlusions and target pose changes.

3.4.4 GOT-10k

GOT-10k [52] is a large-scale dataset with 10,000 videos for training, 180 videos for validation and 180 videos for testing. The main challenge provided by GOT-10k is that there are no duplicated object classes between the training and testing sets. To ensure the fairness of the testing process, we comply with the principle of this benchmark and only use its training set to train our tracker. There are three metrics on GOT-10k, namely AO, $SR_{0.5}$ and $SR_{0.75}$, wherein AO calculates the average overlaps between the predicted bounding box and ground truth. SR measures the percentage of videos that exceeds the preset threshold (e.g., 0.5) of overlaps between the predicted bounding box and ground truth. As shown in the following Table 4, our tracker achieves the best scores for all three metrics (AO, $SR_{0.5}$, $SR_{0.75}$), while DiMP-50 and SiamFC++ rank 2^{nd} and 3^{rd}, respectively. The results on GOT-10k demonstrate the robustness of our tracker, especially its strong generalization ability in handling unseen target classes during the training phase, which matches the demand of generic tracking.

3.4.5 TrackingNet

TrackingNet [55] is a large-scale dataset for object tracking in wild scenes. It has 30,132 videos for training, and 511 videos for testing. It contains more than 14 M dense bounding box annotations of rich categories of classes. The evaluation metrics include precision, normalized precision and success rate (also known as AUC). The precision index measures the percentage of frames whose error of center position is less than the specified threshold, while AUC measures the ratio of frames whose overlap between the predicted target box and the ground-truth box is higher than the specified threshold. To make fair comparisons, we exclude Youtube-BB videos from the training set. As shown in Table 5, we achieve comparable results with SiamRPN++ and is only slightly lag behind DiMP-50. This shows the potentiality of our tracker to be independent of large offline training data.

3.4.6 NFS

NFS [56] contains 100 videos of multiple targets captured by higher speed cameras. Comparative results are listed in

Table 7 Ablation study on historical localization unit

Component	Success	Precision
Fully connected layers	0.578	0.857
HLU	**0.694**	**0.902**

Bold denotes better results in comparison

Table 8 Ablation study on feature fusion module

Component	Success	Precision
Deep cross-correlation	0.589	0.884
Feature fusion module	**0.694**	**0.902**

Bold denotes better results in comparison

Table 9 Ablation study on Gauss–Newton

Component	Success	Precision
Stochastic gradient descent	0.685	0.887
Gauss–Newton	**0.694**	**0.902**

Bold denotes better results in comparison

Table 6 between our tracker and other 11 popular methods. Again, our tracker reaches the highest AUC value. This result shows the robustness of our tracker in capturing fast moving targets with large amplitude of motions. In comparison, some recently published methods (LightTrack [41] and SiamCorners [51]) that are famous for their fast motion extraction abilities are far behind us.

3.5 Analysis of failure cases

We place much emphasis on the ability to capture the target changes. However, we find that our tracker is still prone to interference in dealing with similar targets. As shown in Fig. 9 (the yellow box indicates the ground truth, the red box denotes the tracking results of our model), we originally track one of the wild goose in the first frame, but it is distracted by another similar goose in the following frames, and finally lose the target. The reason for this failure is that we adopt feature concatenation scheme, which reserves features of distractors and therefore could not differentiate similar targets in some scenarios.

3.6 Ablation study

In this section, ablation studies are conducted on OTB100 to verify the effectiveness of the proposed modules in our tracker. For fair comparisons, we use the training set with the same parameter settings.

3.6.1 Historical localization unit (HLU)

In this section, we replace the proposed HLU with fully connected layers and compare the performances with and without HLU, as shown below:

Based on the experimental results, HLU improves the success and precision by 16.7% and 5%, respectively, which proves the effect of HLU.

3.6.2 Feature fusion module (FFM)

In this section, we replace the proposed feature fusion module with popular deep cross-correlation module and compare their success and precision on OTB100, as shown in Table 8. Apparently, compared with deep cross-correlation module, our proposed feature fusion module increases the success and precision by 10.5% and 1.8%, respectively.

3.6.3 Gauss–Newton method

In this section, an ablation experiment is carried out on OTB100 to compare the performance between Gauss–Newton method and the traditional Stochastic Gradient Descent method. Based on the results in Table 9, Gauss–Newton method achieves both higher success and precision.

4 Conclusions

In this paper, we analyze the intrinsic attribute of multi-level optimization method for state estimation in visual object tracking and propose a recurrent optimization scheme based on historical target locations along with four design principles for feature fusion. Specifically, we use ResNet18 as our backbone and deploy only two convolution layers in the classification branch. Our proposed architecture not only simplifies the original multi-level network, but also utilizes the target location information in previous step to facilitate current localization.

Extensive experiments have been conducted on six prevailing benchmarks, in which our tracker exhibits competitive performance against other state-of-the-art approaches.

Data availability The raw/processed data required to reproduce these findings will be shared once this paper has been accepted.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Yi S, Li H, Wang X (2016) Pedestrian behavior modeling from stationary crowds with applications to intelligent surveillance. IEEE Trans Image Process 25(9):4354–4368. https://doi.org/10.1109/TIP.2016.2590322
2. Chen X, Ma H, Wan J, Li B, Xia T (2017) Multi-view 3D object detection network for autonomous driving. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 6526–6534 . https://doi.org/10.1109/CVPR.2017.691
3. Yuan D, Shu X, Liu Q, Zhang X, He Z (2022) Robust thermal infrared tracking via an adaptively multi-feature fusion model. Neural Comput Appl 1–12
4. Wang Y, Wei X, Tang X, Wu J, Fang J (2022) Response map evaluation for RGBT tracking. Neural Comput Appl 34(7):5757–5769
5. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
6. Kiran M, Nguyen-Meidine LT, Sahay R, Cruz RMOE, Blais-Morin L-A, Granger E (2022) Dynamic template selection through change detection for adaptive Siamese tracking. arXiv preprint arXiv:2203.03181
7. Fan H, Lin L, Yang F, Chu P, Deng G, Yu S, Bai H, Xu Y, Liao C, Ling H (2019) Lasot: a high-quality benchmark for large-scale single object tracking. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 5369–5378. https://doi.org/10.1109/CVPR.2019.00552
8. Bertinetto L, Valmadre J, Henriques JF, Vedaldi A, Torr PHS (2016) Fully-convolutional Siamese networks for object tracking. In: Hua G, Jégou H (eds) Computer Vision—ECCV 2016 Workshops. Springer, Cham, pp 850–865
9. Li B, Yan J, Wu W, Zhu Z, Hu X (2018) High performance visual tracking with Siamese region proposal network. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 8971–8980. https://doi.org/10.1109/CVPR.2018.00935
10. Li B, Wu W, Wang Q, Zhang F, Xing J, Yan J (2019) Siamrpn++: evolution of Siamese visual tracking with very deep networks. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4277–4286. https://doi.org/10.1109/CVPR.2019.00441
11. Zhu Z, Wang Q, Li B, Wu W, Yan J, Hu W (2018) Distractor-aware Siamese networks for visual object tracking. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision—ECCV 2018. Springer, Cham, pp 103–119
12. Guo, D., Wang J, Cui Y, Wang Z, Chen S (2020) Siamcar: Siamese fully convolutional classification and regression for visual tracking. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6268–6276. https://doi.org/10.1109/CVPR42600.2020.00630
13. Danelljan M, Bhat G, Khan FS, Felsberg M (2019) Atom: accurate tracking by overlap maximization. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4655–4664. https://doi.org/10.1109/CVPR.2019.00479
14. Li X, Huang L, Wei Z, Nie J, Chen Z (2021) Adaptive multi-branch correlation filters for robust visual tracking. Neural Comput Appl 33(7):2889–2904
15. Yuan D, Chang X, Huang P-Y, Liu Q, He Z (2020) Self-supervised deep correlation tracking. IEEE Trans Image Process 30:976–985
16. Yuan D, Chang X, Li Z, He Z (2022) Learning adaptive spatial-temporal context-aware correlation filters for uav tracking. ACM Trans Multimedia Comput Commun Appl (TOMM) 18(3):1–18
17. Zhou J, Wang P, Sun H (2020) Discriminative and robust online learning for Siamese visual tracking. Proc AAAI Conf Artif Intell 34(07):13017–13024. https://doi.org/10.1609/aaai.v34i07.7002
18. Gao J, Zhang T, Xu C (2019) Graph convolutional tracking. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4644–4654. https://doi.org/10.1109/CVPR.2019.00478
19. Dai K, Zhang Y, Wang D, Li J, Lu H, Yang X (2020) High-performance long-term tracking with meta-updater. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6297–6306. https://doi.org/10.1109/CVPR42600.2020.00633
20. Zhou L, Ding X, Li W, Leng J, Lei B, Yang W (2022) A location-aware Siamese network for high-speed visual tracking. Appl Intell. https://doi.org/10.1007/s10489-022-03636-8
21. Wang G, Luo C, Xiong Z, Zeng W (2019) SPM-tracker: series-parallel matching for real-time visual object tracking. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3638–3647. https://doi.org/10.1109/CVPR.2019.00376
22. Wang Q, Zhang L, Bertinetto L, Hu W, Torr PHS (2019) Fast online object tracking and segmentation: a unifying approach. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 1328–1338. https://doi.org/10.1109/CVPR.2019.00142
23. Voigtlaender P, Luiten J, Torr PHS, Leibe B (2020) Siam R-CNN: visual tracking by re-detection. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6577–6587. https://doi.org/10.1109/CVPR42600.2020.00661
24. Cheng S, Zhong B, Li G, Liu X, Tang Z, Li X, Wang J (2021) Learning to filter: Siamese relation network for robust tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4421–4431
25. Tan K, Xu T-B, Wei Z (2022) Imsiam: IOU-aware matching-adaptive Siamese network for object tracking. Neurocomputing 492:222–233. https://doi.org/10.1016/j.neucom.2022.04.003
26. Zhang Z, Peng H (2019) Deeper and wider Siamese networks for real-time visual tracking. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 4586–4595. https://doi.org/10.1109/CVPR.2019.00472
27. Chen Z, Zhong B, Li G, Zhang S, Ji R (2020) Siamese box adaptive network for visual tracking. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6667–6676 . https://doi.org/10.1109/CVPR42600.2020.00670
28. Zhang Z, Peng H, Fu J, Li B, Hu W (2020) Ocean: object-aware anchor-free tracking, pp 771–787. https://doi.org/10.1007/978-3-030-58589-1_46
29. Fan H, Ling H (2019) Siamese cascaded region proposal networks for real-time visual tracking. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7944–7953. https://doi.org/10.1109/CVPR.2019.00814
30. Saribas H, Cevikalp H, Köpüklü O, Uzun B (2022) Trat: tracking by attention using spatio-temporal features. Neurocomputing 492:150–161. https://doi.org/10.1016/j.neucom.2022.04.043
31. Xu Y, Wang Z, Li Z, Yuan Y, Yu G (2020) Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. Proc AAAI Conf Artif Intell 34(07):12549–12556. https://doi.org/10.1609/aaai.v34i07.6944
32. Bhat G, Danelljan M, Van Gool L, Timofte R (2019) Learning discriminative model prediction for tracking. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 6181–6190. https://doi.org/10.1109/ICCV.2019.00628
33. Danelljan M, Van Gool L, Timofte R (2020) Probabilistic regression for visual tracking. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 7181–7190. https://doi.org/10.1109/CVPR42600.2020.00721

34. Chen S, Qiu C, Zhang Z (2022) An efficient method for tracking failure detection using parallel correlation filtering and Siamese network. Appl Intell 52(7):7713–7722. https://doi.org/10.1007/s10489-021-02768-7
35. Zhou Y, Zhang Y (2022) Siamet: a Siamese based visual tracking network with enhanced templates. Appl Intell 52(9):9782–9794. https://doi.org/10.1007/s10489-021-03057-z
36. Wu Y, Lim J, Yang M-H (2015) Object tracking benchmark. IEEE Trans Pattern Anal Mach Intell 37(9):1834–1848. https://doi.org/10.1109/TPAMI.2014.2388226
37. Li P, Chen B, Ouyang W, Wang D, Yang X, Lu H (2019) Gradnet: gradient-guided network for visual object tracking. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 6161–6170 . https://doi.org/10.1109/ICCV.2019.00626
38. Zhang L, Gonzalez-Garcia A, Weijer JVD, Danelljan M, Khan FS (2019) Learning the model update for Siamese trackers. In: 2019 IEEE/CVF international conference on computer vision (ICCV), pp 4009–4018. https://doi.org/10.1109/ICCV.2019.00411
39. Blatter P, Kanakis M, Danelljan M, Van Gool L (2021) Efficient visual tracking with exemplar transformers. arXiv preprint arXiv:2112.09686
40. Chen X, Wang D, Li D, Lu H (2022) Efficient visual tracking via hierarchical cross-attention transformer. arXiv preprint arXiv:2203.13537
41. Yan B, Peng H, Wu K, Wang D, Fu J, Lu H (2021) Lighttrack: finding lightweight neural networks for object tracking via one-shot architecture search. In: 2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 15175–15184. https://doi.org/10.1109/CVPR46437.2021.01493
42. Zhou L, Ding X, Li W, Leng J, Lei B, Yang W (2022) A location-aware Siamese network for high-speed visual tracking. Appl Intell 1–17
43. Gao L, Liu B, Fu P, Xu M, Li J (2022) Visual tracking via dynamic saliency discriminative correlation filter. Appl Intell 52(6):5897–5911
44. Kristan M, Leonardis A, Matas J, Felsberg M (2019) The sixth visual object tracking vot2018 challenge results. In: Leal-Taixé L, Roth S (eds) Computer Vision—ECCV 2018 Workshops. Springer, Cham, pp 3–53
45. Sun C, Wang D, Lu H, Yang M-H (2018) Correlation tracking via joint discrimination and reliability learning. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 489–497. https://doi.org/10.1109/CVPR.2018.00058
46. Bhat G, Johnander J, Danelljan M, Khan FS, Felsberg M (2018) Unveiling the power of deep tracking. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision—ECCV 2018. Springer, Cham, pp 493–509
47. Xu T, Feng Z, Wu X-J, Kittler J (2021) Adaptive channel selection for robust visual object tracking with discriminative correlation filters. Int J Comput Vis. https://doi.org/10.1007/s11263-021-01435-1
48. Luo Y, Xiao H, Ou J, Chen X (2022) Siamsmdfff: Siamese network tracker based on shallow-middle-deep three-level feature fusion and clustering-based adaptive rectangular window filtering. Neurocomputing 483:160–170. https://doi.org/10.1016/j.neucom.2022.02.027
49. Danelljan M, Bhat G, Khan FS, Felsberg M (2017) Eco: efficient convolution operators for tracking. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 6931–6939. https://doi.org/10.1109/CVPR.2017.733
50. Nam H, Han B (2016) Learning multi-domain convolutional neural networks for visual tracking. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 4293–4302. https://doi.org/10.1109/CVPR.2016.465
51. Yang K, He Z, Pei W, Zhou Z, Li X, Yuan D, Zhang H (2022) Siamcorners: Siamese corner networks for visual tracking. IEEE Trans Multimedia 24:1956–1967. https://doi.org/10.1109/TMM.2021.3074239
52. Huang L, Zhao X, Huang K (2021) Got-10k: a large high-diversity benchmark for generic object tracking in the wild. IEEE Trans Pattern Anal Mach Intell 43(5):1562–1577. https://doi.org/10.1109/TPAMI.2019.2957464
53. Yang T, Xu P, Hu R, Chai H, Chan AB (2020) Roam: recurrently optimizing tracking model. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 6717–6726 . https://doi.org/10.1109/CVPR42600.2020.00675
54. Xing D, Evangeliou N, Tsoukalas A, Tzes A (2022) Siamese transformer pyramid networks for real-time UAV tracking. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 2139–2148
55. Müller M, Bibi A, Giancola S, Alsubaihi S, Ghanem B (2018) Trackingnet: a large-scale dataset and benchmark for object tracking in the wild. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision—ECCV 2018. Springer, Cham, pp 310–327
56. Galoogahi HK, Fagg A, Huang C, Ramanan D, Lucey S (2017) Need for speed: a benchmark for higher frame rate object tracking. In: 2017 IEEE international conference on computer vision (ICCV), pp 1134–1143. https://doi.org/10.1109/ICCV.2017.128
57. Danelljan M, Robinson A, Khan F, Felsberg M (2016) Beyond correlation filters: learning continuous convolution operators for visual tracking
58. Qi Y, Zhang S, Qin L, Huang Q, Yao H, Lim J, Yang M-H (2019) Hedging deep features for visual tracking. IEEE Trans Pattern Anal Mach Intell 41(5):1116–1130. https://doi.org/10.1109/TPAMI.2018.2828817
59. Wang L, Ouyang W, Wang X, Lu H (2015) Visual tracking with fully convolutional networks. In: 2015 IEEE international conference on computer vision (ICCV), pp 3119–3127. https://doi.org/10.1109/ICCV.2015.357