Neurocomputing 492 (2022) 222–233

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

IMSiam: IoU-aware Matching-adaptive Siamese network for object tracking

Ke Tan, Ting-Bing Xu, Zhenzhong Wei *

Key Laboratory of Precision Opto-mechatronics Technology, Ministry of Education, Beihang University, No. 37 Xueyuan Rd. Haidian District, 100191 Beijing, China

ARTICLE INFO

Article history:
Received 8 December 2020
Revised 26 February 2022
Accepted 3 April 2022
Available online 7 April 2022

Keywords:
Object tracking
Siamese networks
IoU-aware
Matching-adaptive

ABSTRACT

Fully convolutional Siamese networks have shown their advantages in the visual object tracking task. However, most existing Siamese-based trackers still suffer from poor matching information and a low correlation between the classification score and bounding box estimation. To address these issues, we propose an IoU-aware Matching-adaptive Siamese network (IMSiam) for visual tracking in this paper. Specifically, a matching-adaptive network is proposed, which integrates multiple types of encoded feature maps and adaptively sample matching information to simultaneously perform target classification and bounding box regression. And we introduce an IoU-aware head to equip the Siamese detector with the capacity of IoU prediction for each regressed box, which then penalizes the classification score to obtain a more confident target region. Benefiting from the adaptive matching information and IoU-based classification outputs, the model can localize the target more accurately while avoiding the redundant search of scale penalty factor during tracking. Experimental results on OTB2013, OTB2015, VOT2018, VOT2019, LaSOT, and Got10k datasets demonstrate that the proposed tracker performs favorably against state-of-the-art methods.

1. Introduction

The goal of visual object tracking is to localize the tracked object in the subsequent frames of a video given the initial target annotation in the first frame. It has drawn widespread attention so far and has been widely applied in various areas, such as self-driving, security monitoring, human–computer interactions, augmented reality and intelligent transportation [1,2]. However, due to many factors including occlusion, scale variation, etc. [3], designing a real-time high-performance tracker is still a very challenging task [4].

Inspired by cognitive studies of Siamese networks in similarity learning, visual tracking methods have made substantial progress recently. One of the early works is the SiamFC tracker [8], which uses a fully convolutional Siamese network to learn a general similarity metric by cross-correlation between the embedding of the target template and search image. Later the SiamRPN tracker [9] introduces the region proposal network (RPN) [10] after the Siamese network to perform the joint training of classification and box regression based on the cross-correlation feature maps. It achieves good accuracy while running over 160 FPS on multiple tracking benchmarks [3,11]. Subsequently, many RPN-based Siamese trackers are proposed, such as DaSiamRPN [12], SiamRPN++ [13], DWSiamRPN [14], CSiamRPN [15], and SPM [16], which greatly improve tracking performance. However, the above-mentioned anchor-based trackers need to design multi-scale anchors manually, which hurts the generalization to different tracking tasks. Inspired by the anchor-free approaches (such as FCOS [17]) in object detection, more recent trackers directly predict the category probabilities of all grid centers and regress the boundary distance. The representative works such as SiamFC++ [7], SiamBAN [6], and SiamCAR [5], can achieve state-of-the-art results while keeping a simple and effective framework. However, these trackers are still easy to fail in some complex tracking scenes, as is shown in Fig. 1.

According to the observation, we notice that the existing Siamese trackers usually only use cross-correlation feature maps to perform one-shot detection and possess a low correlation between the results of classification task and box regression task. Firstly, the matching information encoded by the cross-correlation is limited. Due to the loss of the similarity measure on the absolute scale, Siamese correlation-based trackers easily drift to the distractor more salient than the tracking object. As is shown in Fig. 1Basketball), some Siamese trackers [5,6] mistakenly detect the players in white. To address this problem, we propose a novel matching-adaptive

* Corresponding author.
E-mail addresses: tanke@buaa.edu.cn (K. Tan), tingbing_xu@buaa.edu.cn (T.-B. Xu), zhenzhongwei@buaa.edu.cn (Z. Wei).

https://doi.org/10.1016/j.neucom.2022.04.003
0925-2312/

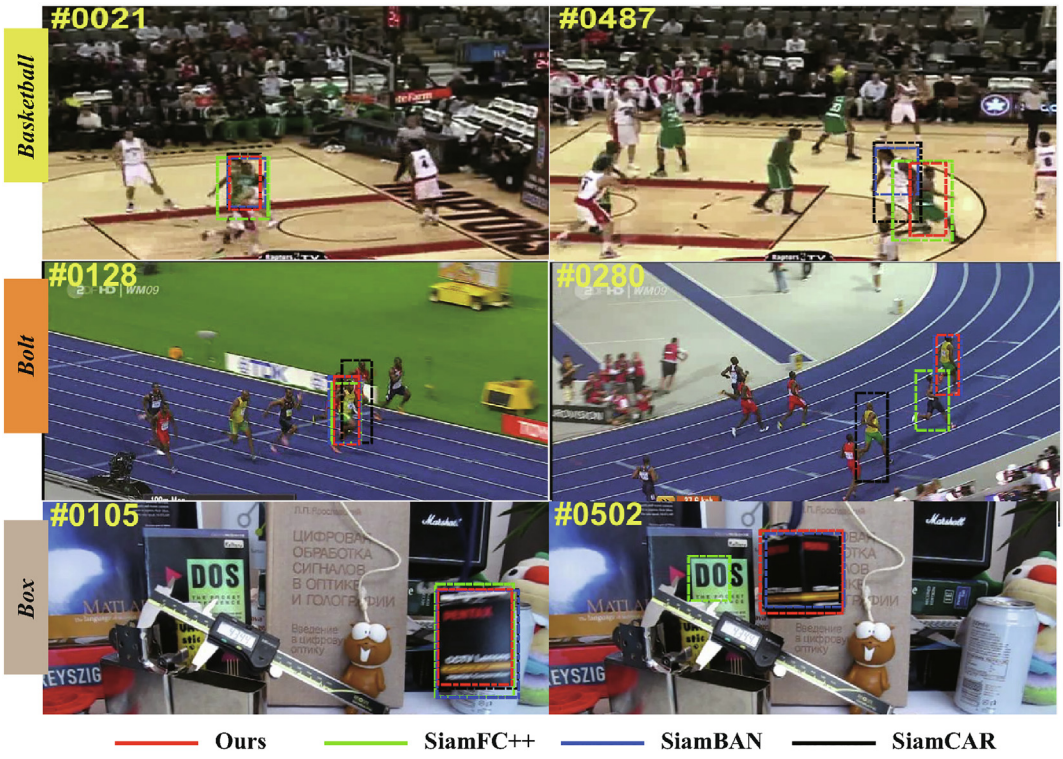


Fig. 1. Proposed IoU-aware Matching-adaptive Siamese Network for visual tracking, which is fully anchor-free and have less hyper-parameters during tracking. It performs favorably against existing anchor-free siamese trackers, such as SiamCAR [5], SiamBAN [6] and SiamFC++ [7]. Visual results of different methods on the OTB dataset are shown here. Best viewed in color.

network to fuse multi-type feature maps and adaptively extract matching information for different tasks. Different from the existing Siamese-based trackers [6,7] that only use cross-correlation feature maps to perform classification and regression, we concatenate three types of feature maps, which are the L1 difference between the embedding of target template and search image, the search identity feature maps and the padded depth-wise cross-correlation feature maps. The L1 difference and search identity feature maps can complement to the cross-correlation maps, which are critical for matching information embedding. The resulting fused maps are then shared by two parallel downsampling layers which are used to adaptively extract separate matching features, forming the basis for subsequent heads.

Secondly, the mismatch between classification scores and the qualities of estimated bounding boxes leads to a decrease in tracking accuracy, as is shown in Fig. 2. To solve this problem, we propose an IoU-aware head parallel with the bounding box regression head, which can predict the IoU score between the predicted bounding box and the ground truth. The IoU scores can serve as the quality assessment of the regressed bounding boxes. Unlike the existing IoU-aware network, such as ATOM that refine the target box by maximizing the IoU with several gradient ascent iterations to generate better results, we use the predicted IoU scores to penalize the classification scores for obtaining a more confident score map during tracking. Benefit from this module, we need not manually search the scale change penalty factor to re-rank the classification scores of every predicted target box to get the best one. It is also a key reason why most existing trackers [6,7,5,9,13] have the worse generalization ability to different tracking scenes.

Based on the two ideas mentioned above, we design an IoU-aware Matching-adaptive Siamese network (IMSiam) for visual tracking. More precisely, IMSiam is an end-to-end anchor-free network framework, which consists of three subnetworks: the Siamese backbone network, the matching-adaptive neck network and the IoU-aware head network. Due to the anchor-free manner, we do not need to design anchors manually. And adaptively sampling matching information for classification and regression

Fig. 2. Illustration for the problem of mismatch between the classification scores and bounding box estimates. The yellow boxes are the ground truth. The black estimated bounding boxes (denoted as A) have higher classification scores but lower IoU scores than the red bounding boxes (denoted as B). Best viewed in color.

branches drives the model to learn more discriminative features for different tasks. In the tracking phase, thanks to the IoU-aware head, we also do not need to search for the scale change penalty factor, which avoids the tricky hyper-parameter selection. And there are only two tracking hyper-parameters: the displacement and size smooth weights, making our tracker more robust to different challenging scenes. We evaluate the proposed tracker, dubbed as IMSiam, on OTB2013 [18], OTB2015 [3], VOT2018 [19], VOT2019 [4], LaSOT [20], and Got10k [21] datasets. It achieves state-of-the-art performance while running in real-time. The code will be released at https://github.com/tankebuaa/pyvis.

The main contributions of this work are listed below in threefold:

- We propose a matching-adaptive network to fuse multi-type feature maps and adaptively extract matching information for performing classification and regression tasks simultaneously.
- We design a dedicated IoU head to predict the IoU between the regressed box and the ground truth. In the tracking stage, it can be used to re-rank the classification scores, which helps avoid searching for the scale change penalty factor and makes the model neater.
- We design a novel IoU-aware Matching-adaptive Siamese tracker that can be trained end-to-end offline in an anchor-free manner. Experiments show that our IMSiam tracker achieves state-of-the-art performance while running in real-time on six popular tracking datasets.

The rest of this paper is organized as follows. Section 2 gives an overview of prior works most relevant to this work. Section 3 details the proposed method. In Section 4, we provide the implementations and the results of performed experiments. Conclusions are finally given in Section 5.

2. Related work

In the recent tracking community, significant attention has been paid to the methods based on the convolutional neural networks (CNN). Among them are two dominant tracking strategies. One is that CNN is treated as an offline feature extraction step combined with an online object appearance classifier. These methods show significant advantages over trackers using hand-crafted features, such as HOG and CN [22]. CNN-SVM [23] combines a pre-trained CNN and online SVMs to obtain target-specific saliency maps for visual tracking. DeepSRDCF [24] and HCF [25] adopt deep features from pre-trained CNN to boost the performance of discriminative correlation filter (DCF) framework [26,27]. Then the subsequent DCF-based trackers use the multi-level deep features, such as CCOT [28], ECO [29], DeepSTRCF [30] and UPDT [31]. Although these trackers can achieve good accuracy, they usually run very slow and cannot meet the real-time requirements.

The other strategy is to integrate CNN into a network model that can directly output tracking results. DeepTrack [32] proposes a target-specific CNN, which is trained incrementally during tracking with new examples obtained online. MDNet [33] exploits a multi-domain CNN to learn the shared representation of objects from different sequences. TCNN [34] maintains multiple layer-shared CNNs in a tree structure to learn ensemble models. CREST [35] learns a base and residual network on a single convolutional layer. MRCNN [36] trains a manifold regularized deep CNN model with both labeled and unlabeled data online and performs tracking under the framework of particle filters. DSLT* [37] learns a deep regression tracker with shrinkage loss, which achieves good generalization performances. The above algorithms need costly stochastic gradient descent (SGD) updating, which hurts their running efficiency. To this end, some methods employ powerful optimization strategies to ensure rapid convergence such as ATOM [38], DiMP [39], and PrDiMP [40]. However, their initialization step still costs much time.

In addition to the above methods, trackers based on the offline-trained Siamese network have also been well studied due to their end-to-end training capabilities and high efficiency. As one of the pioneering works, SiamFC [8] constructs a fully convolution Siamese network and introduces the cross-correlation layer to combine feature maps. Owing to its light structure and no need to update, SiamFC can run efficiently at 86 FPS. Afterwards, there exists a large number of follow-up work of SiamFC. CFNet [41] introduces a correlation filter layer to update the model. Triplet loss is then used to train SiamFC and CFNet, which brings a performance boost [42,43]. DQL [44,45] exploits continuous deep Q-learning to adjusts hyperparameters for different sequences to achieve the best accuracy. SA-Siam [46] introduces a twofold Siamese network to exploit complementary features for visual tracking. LSSiam [47] extracts the local semantic features to solve the drift problem of SiamFC. RASNet [48] and HASN [49] incorporate different types of attention mechanisms into the Siamese network, respectively. Notably, SiamRPN [9] combines the Siamese network with the region proposal network, which performs joint classification and regression for visual tracking. DaSiamRPN [12] further makes use of hard negative sample training to improve discriminative ability. SiamRPN++ [13] successfully uses the deeper CNN like ResNet50 as the backbone by random shift, which achieves better tracking accuracy than trackers with shallow backbone networks. More recently, CLNet [50] makes full use of the sequence-specific samples to train SiamRPN++, achieving promising improvement. However, these RPN-based trackers need to carefully design the hyper-parameters of the anchors, such as the aspect ratio and the numbers. Aims to this limitation, many anchor-free trackers are proposed, such as SiamFC++ [7], SiamCAR [5] and SiamBAN [6]. We note that all these trackers combine the Siamese network, the cross-correlation layer and the heads from FCOS [17], and use the scale change penalty [13] to refine the classification score. In this paper, we also propose an anchor-free Siamese model. Differently from existing anchor-free methods, we design a matching-adaptive module and an IoU-aware head. The former aims to adaptively sample the matching feature for classification and regression tasks, while the latter aims to replace the scale change penalty to refine the classification score.

3. The proposed method

In this section, we propose an IoU-aware Matching-adaptive Siamese network for visual tracking. The overall architecture is shown in Fig. 3, which consists of three subnetworks: the Siamese backbone network, the Matching-adaptive neck network, and the IoU-aware head network. The Siamese backbone network is utilized to extract the features of the target template z and the search image x. Then, feature maps from each stage are fed into the following neck network, which outputs the matching-adaptive feature maps for subsequent tasks. Afterwards, an IoU-aware head network is applied to simultaneously perform classification and localization based on the matching-adaptive feature maps. In the following, we will detail the network design, the training and tracking procedures.

3.1. Siamese backbone network

We adopt the ResNet50 [51] with atrous convolution to develop our Siamese backbone network. As is shown in Fig. 3, it comprises two identical branches, which are used to extract features of the

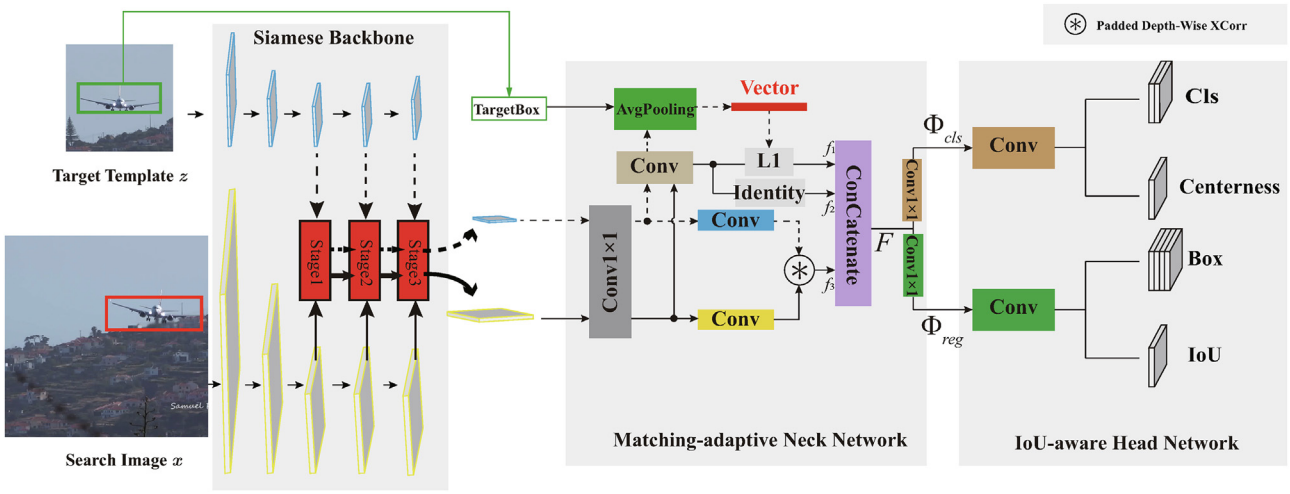


Fig. 3. An illustration of our proposed network framework. For the feature maps in each stage, we use a matching-adaptive neck subnetwork and an IoU-aware head subnetwork to predict the target location and shape. The matching-adaptive subnetwork takes template features and search features as inputs, and computes the adaptive feature maps for the classification and regression modules in the IoU-aware head subnetwork, respectively.

target template z and the search image x. Specifically, the backbones in two branches share the same parameters. With this design, it ensures the same transformation applied to both z and x, which is crucial for similarity metric learning. Besides, we adopt different atrous rates in the last three stages while keeping the spatial downsample stride $s = 8$ as in SiamRPN++ [13]. Features at different stages in ResNet50 contain different information. Low-level features consist of spatial cues, while high-level features output more semantic information. Therefore, we utilize the features from stage 1 to stage 3 to improve the robustness of the model. Next, we simply perform the same procedures at each stage independently.

3.2. Matching-adaptive neck network

Different from object detection which usually assumes the categories of the objects are predefined, visual tracking typically needs to encode the target appearance information into the framework. Most recent methods calculate the depth-wise cross-correlation between the feature representations learned from the target template and the search image as the basis for the subsequent tasks. Some methods, such as SiamRPN++ [13], SiamFC++ [7] and SiamBAN [6], use two separate depth-wise cross-correlation branches to predict the classification scores and regression results. The SiamCAR tracker [5] only uses one depth-wise cross-correlation layer to produce similarity maps, which are then used by the classification and regression heads. However, the limited similarity information only from the cross-correlation layer still restricts its tracking performances. Here we design a shared matching-adaptive neck network to extract the features for the subsequent heads.

The matching-adaptive neck network firstly concatenates three types of feature maps: the L1 difference, the search identity and the padded depth-wise cross-correlation. Specifically, we firstly add a 1×1 convolution to downsample the input feature channel to c (we set $c = 256$ in this paper). Based on the resulting feature maps from template and search branches, we take the following steps to obtain the desired multi-type feature maps, as illustrated in Fig. 4.

3.2.1. The L1 difference maps

We first adopt a shared 3×3 Conv-BN-ReLU block to adjust features from template and search branches. Then an average pooling layer is used to pool the target region in template feature maps to an embedding vector with the annotated target box, as is shown in Fig. 5. Formally, we denote the search feature map is f_x and the template feature map is f_z with the target box $\mathscr{B} = (X_1, Y_1, X_2, Y_2)$, where $(X_1, Y_1) and (X_2, Y_2)$ are the coordinates of the top-left and bottom-right corners in the template image. Then we project the two corner points to f_z, obtaining the continuous coordinates of corners (x_1, y_1) and (x_2, y_2). We define the average pooling operation by computing a two-order integral:

$$avgpool(f_z, \mathscr{B}) = \frac{\int_{y_1}^{y_2} \int_{x_1}^{x_2} f_z(x,y)dxdy}{(x_2 - x_1) \times (y_2 - y_1)}. \quad (1)$$

Note that the above operation is continuously differentiable [52]. Finally, we compute the absolute difference between the template embedding vector $v = avgpool(f_z, \mathscr{B})$ and the feature map of the search image f_x at each spatial position. The output L1 difference maps denoted as $f_1 \in \mathbb{R}^{c \times w \times h}$ is calculated by

$$f_1 = |f_x - v|. \quad (2)$$

3.2.2. The search identity feature maps

We just take the adjusted search feature maps in the L1 difference branch as the encoded identity feature maps of the search image, which is denoted as $f_2 \in \mathbb{R}^{c \times w \times h}$.

3.2.3. The padded depth-wise cross-correlation maps

We crop a $d \times d$ ($d = 7$ in this paper) center region of the target template features and then adopt a non-shared 3×3 Conv-BN-ReLU block to adjust features from template and search branches [13]. The resulting feature maps are finally combined by a padded depth-wise cross-correlation layer (PDW-XCorr), where the template feature maps are used as a convolution kernel and the search feature maps are padded with zeros to keep the resolution unchanged. The output feature maps are denoted as $f_3 \in \mathbb{R}^{c \times w \times h}$.

Then we can obtain the fused feature maps $F \in R^{(3c) \times w \times h}$ by

$$F = concat(f_1, f_2, f_3), \quad (3)$$

where $concat(\cdot)$ denotes the concatenation operation along the channels.

Since it aims to leverage F to perform classification and bounding box regression, we adaptively sample the features from it for different tasks. we define the extraction of matching features as a function φ by

$$\Phi = sample\{f_1, f_2, f_3\}, = \varphi(F), \quad (4)$$

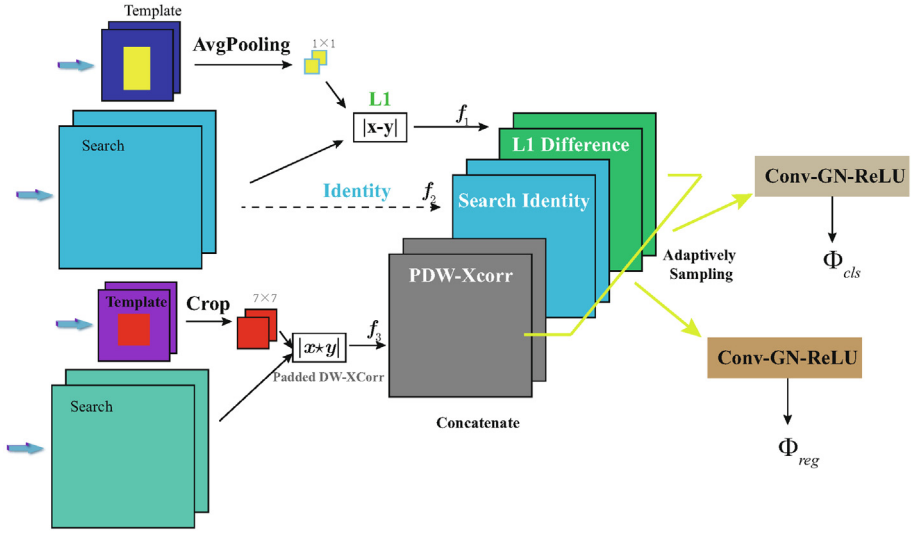


Fig. 4. An illustration of our proposed matching-adaptive network.

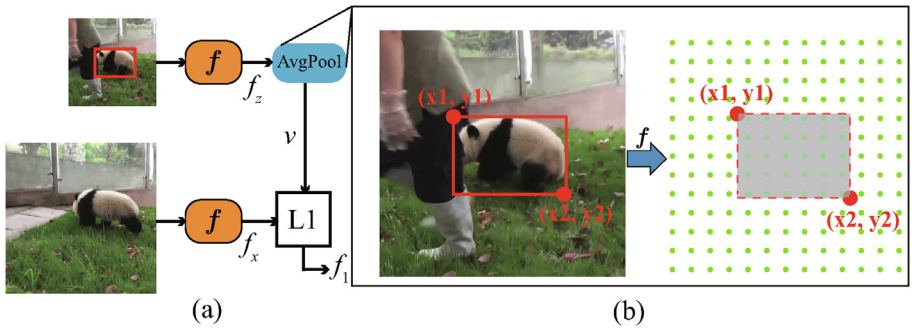


Fig. 5. (a) The overall network architecture of the L1 difference feature extraction between the template and search image. (b) The illustration of the average pooling operation based on the target region.

where Φ is the obtained matching features. Here we model the function φ by two 1×1 Conv-GN-ReLU blocks to extract separate features for classification and regression branches, denoted as Φ_{cls} and Φ_{reg} respectively, as is shown in Fig. 4. And *GN* means Group Normalization.

3.3. IoU-aware head network

We adopt an anchor-free scheme to design the head network, which works as follows. Given a target template z and a search image x, the location and the shape of the target can be represented by a 6-tuple in the form of (i,j,l,t,r,b), where (i,j) is the spatial point coordinate in x and (l,t,r,b) are the distances to the four boundaries. Then the location and shape can be considered to follow a distribution as

$$p(i,j,l,t,r,b|z,x) = p(i,j|z,x)p(l,t,r,b|i,j,z,x), \quad (5)$$

$$\propto p(i,j|\Phi_{cls})p\left(l,t,r,b|i,j,\Phi_{reg}\right). \quad (6)$$

From this factorization, we have two important intuitions. Firstly, the two items correspond to the classification and regression tasks respectively, so that we use two separate branches to perform the corresponding task. Secondly, to maximize the probability we assign quality assessments to both classification and regression results. We notice that most methods directly use the anchor-free detection head from FCOS [17], such as SiamFC++ [7], SiamBAN [6] and SiamCAR [5], which use the centerness metric [17] to assess the classification result. To assess the regression result, they follow SiamRPN++ [13] which uses the previous bounding box to compute a scale change penalty by a manually-designed function. This not only introduces tricky hyper-parameters but also hurts the adaption abilities in different tracking scenes. To solve this problem, our key idea is to learn to predict the IoU scores for the regressed boxes by adding an IoU head parallel with the regression head, which is then exploited to penalize the classification scores. With this simple design, it can improve the accuracy of classification and localization results.

As shown in Fig. 3, the IoU-aware head network consists of a classification module and a regression module, which receive the feature maps Φ_{cls} and Φ_{reg} respectively. The resulting features in each module are then adjusted by a set of three 33 Conv-GN-ReLU blocks respectively. For the classification module, we use a single 33 convolution layer to output a two-channel score map $C \in \mathbb{R}^{2\times w\times h}$ for foreground-background classification. We also use another single 33 convolution layer parallel to the classification head to output a one-channel center-ness map $P \in \mathbb{R}^{1\times w\times h}$. For the regression module, we use a single 33 convolution layer to output a four-channel regression map. Because the regression targets are always positive, we also employ $\exp(x)$ to map the regression output number to $(0,\infty)$, denoted as $R \in \mathbb{R}^{4\times w\times h}$. Last, we adopt a single 33 convolution layer to output a one-channel map $I \in \mathbb{R}^{1\times w\times h}$ for the IoU prediction between the regressed bounding box and the ground-truth. This can be seen as a quality estimation of each regressed box and provides a weight for the corresponding

classification score. Next, we will detail the training labels and loss function.

3.4. Training labels and loss function

The proposed model receives a pair of images as input. Follow the practice of SiamRPN++ [13], we generate a target template z and a search image x containing the same tracking object as one training sample. Let $B = (cx, cy, w, h)$ be the ground-truth target bounding box in the search image, where (cx, cy) is the target center, w and$h$ is the target width and height respectively.

3.4.1. Classification labels

For each location $p_c = (i,j)$ on the classification score map, we can map it back on to the search image as p which is near the center of the receptive field. We then define three types of regions for each location p.

(1)The center region $C\mathscr{R} = (cx, cy, \Delta_1 w, \Delta_1 h)$ defines the center part of the target box B, where $0 < \Delta_1 < 1$ and we set $\Delta_1 = 0.75$ in this work. Each location p falls into $C\mathscr{R}$ is assigned as a positive sample.
(2)The ignore region $I\mathscr{R} = (cx, cy, \Delta_2 w, \Delta_2 h) \setminus C\mathscr{R}$ is a larger region excluding $C\mathscr{R}$, where $(\Delta_1 < \Delta_2 \leqslant 1)$ and we set $\Delta_2 = 1$ in this work. Each location p falls into $I\mathscr{R}$ is marked as "ignore" and excluded during training.
(3)The outside region $O\mathscr{R}$ is the whole search image excluding $C\mathscr{R}$ and $I\mathscr{R}$. Each location p falls into $O\mathscr{R}$ is assigned as a negative sample. So that we set the class label at the location (i,j) by

$$C^*(:,i,j) = \begin{cases} [1,0] & (i,j) \in C\mathscr{R}, \\ [-1,-1] & (i,j) \in I\mathscr{R}, \\ [0,1] & (i,j) \in O\mathscr{R}, \end{cases} \tag{7}$$

where the element -1 means "ignore".

3.4.2. Regression labels

We set a regression target for each point (i,j) in the regression map R^* by its project location (x,y) in the search image, which contains a 4D real vector $R^*(:,i,j) = [l,t,r,b]$ calculated by

$$\begin{cases} l = x - (cx - w/2), \\ t = y - (cy - h/2), \\ r = (cx + w/2) - x, \\ b = (cy + h/2) - y. \end{cases} \tag{8}$$

3.4.3. Center-ness labels

To suppress the low-quality predicted bounding boxes that are far away from the center of the ground-truth, we assign center-ness scores for all positive samples as in FCOS [17]. Given the regression targets $[l,t,r,b]$ for a location (i,j) in the center-ness map, its corresponding center-ness target is calculated by

$$P^*(:,i,j) = \left[\sqrt{\frac{\min(l,r)}{\max(l,r)} \times \frac{\min(t,b)}{\max(t,b)}} \right]. \tag{9}$$

3.4.4. IoU labels

Given the results at the location (i,j) in the regression map, we can obtain a predicted bounding box $\widehat{B}_{i,j}$. Then we assign an IoU label for the IoU prediction score at the location (i,j) by

$$I^*(:,i,j) = \left[\frac{\widehat{B}_{i,j} \cap B}{\widehat{B}_{i,j} \cup B} \right]. \tag{10}$$

3.4.5. Loss function

We calculate the loss at each stage independently. For convenience, we use an index $l (l \in \{1,2,3\})$ to denote the l-th stage. Given the output of the classification head C^l, we compute the class loss by cross-entropy which can express as

$$\mathscr{L}^l_{cls} = CrossEntropy\left(C^*, C^l\right). \tag{11}$$

And given the box regression result R^l, we adopt the IoU loss function to calculate the regression loss by

$$\mathscr{L}^l_{reg} = -\log\left(IoU\left(R^*, R^l\right)\right), \tag{12}$$

where IoU means the area ratio of the intersection over the union of the two boxes. Besides, the binary cross-entropy (BCE) is adopted to calculate the loss of both center-ness and IoU heads, which are defined as follows

$$\mathscr{L}^l_{cen} = BCE\left(P^*, P^l\right), \tag{13}$$

$$\mathscr{L}^l_{iou} = BCE\left(I^*, I^l\right). \tag{14}$$

We explore three different level features from stages 1,2,3 of ResNet50 to perform predictions. Finally, we average the three-stage loss as the overall training loss function:

$$\mathscr{L} = \frac{1}{3}\sum_{l=1}^{3}\left(\mathscr{L}^l_{cls} + \lambda_1 \mathscr{L}^l_{reg} + \lambda_2 \mathscr{L}^l_{cen} + \lambda_3 \mathscr{L}^l_{iou}\right), \tag{15}$$

where $\lambda_1, \lambda_2, \lambda_3$ are the weight factors to balance different tasks and l is the stage index. In this paper, we simply set $\lambda_1 = 1.2, \lambda_2 = \lambda_3 = 0.5$ empirically.

Algorithm 1 Tracking algorithm for IMSiam

Require: Sequence I, initial bounding box B_0, Pre-trained network, cosine window weight λ, size interpolation factor γ.
Ensure: Target bounding boxes of all images B.
1: Compute sequence length N and set image index $i = 0$.
2: **while** $i < N$ **do**
3: Load current image I_i.
4: **if** $i == 0$ **then**
5: Crop the template patch z and initialize the network.
6: Initialize the cosine window w.
7: **else**
8: Crop the search patch x.
9: Feed x into the model and output four prediction maps C^l, P^l, R^l, I^l for each stage l.
10: Compute the final classification score map by Eqs. (16) and (17) with the cosine window weight λ.
11: Compute the target bounding box by Eq. (18).
12: Update target size by Eq. (19) with the linear interpolation factor γ and obtain the final bounding box B_i.
13: **end if**
14: Update image index $i = i + 1$.
15: **end while**
16: **return** B

3.5. Tracking

Once the proposed model is trained offline, it is simply evaluated once per frame without any adaption and only has two hyper-parameters during tracking phase. Given the first frame with annotation, we firstly extract the target template to initialize

the model. Since we do not update the template and the network, the feature maps related to the template only need to be calculated once. During the tracking phase, we extract a larger region at the previous target location as the search image. The aim is to predict the bounding box for the target. To this end, we feed the search image into the network, which outputs four prediction maps C^l, P^l, R^l, I^l for each stage l. To get the best bounding box for the target, the center-ness score map and the IoU score map are used to penalize the classification map which produces the refined classification map by

$$\widehat{C}^l = C^l[0] \odot P^l \odot I^l, \tag{16}$$

where $\odot$denotes the element-wise multiplication. Then we merge the multi-stage prediction and add a cosine window to suppress the large displacement. The final classification map is obtained by

$$C = \frac{1-\lambda}{3}\sum_{l=1}^{3}\widehat{C}^l + \lambda w, \tag{17}$$

where w is the cosine window weight and λdenotes the balance weight. The highest score in C indicates the most confident target location (i,j), and we query the regression maps at this location. The final bounding box is calculated by average the regression results of all stages by

$$R = \frac{1}{3}\sum_{l=1}^{3}R^l. \tag{18}$$

Furthermore, we update the target size by linear interpolation γ to keep the shape changing smoothly:

$$\begin{bmatrix} w \\ h \end{bmatrix} \Leftarrow (1-\gamma)\begin{bmatrix} w\prime \\ h\prime \end{bmatrix} + \gamma\begin{bmatrix} w \\ h \end{bmatrix}, \tag{19}$$

where w and h are the width and height of the final bounding box, the superscript $(\cdot)'$ means the state in the last frame. Algorithm 1 illustrates the tracking procedure of the proposed IMSiam approach.

4. Experiments

4.1. Implementation details

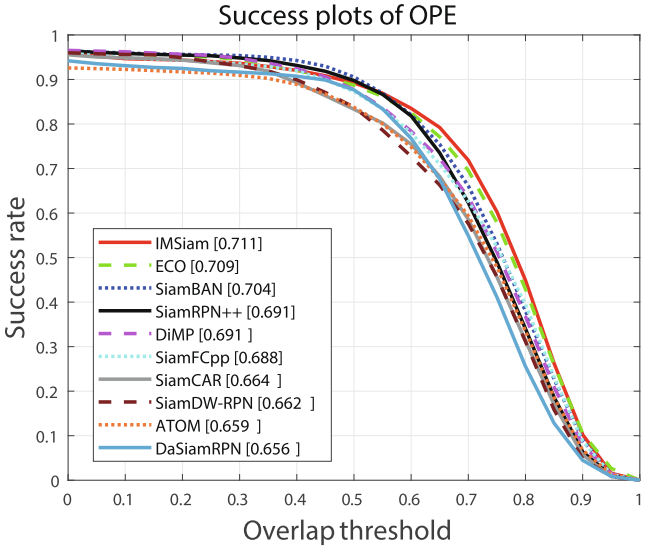
Our approach is implemented in Python3.7 with the PyTorch1.4 deep learning framework. The model is trained on a workstation with Intel Xeon Gold 6246 CPU, 256 GB RAM and 4 NVIDIA RTX2080Ti GPUs.

4.1.1. Training details

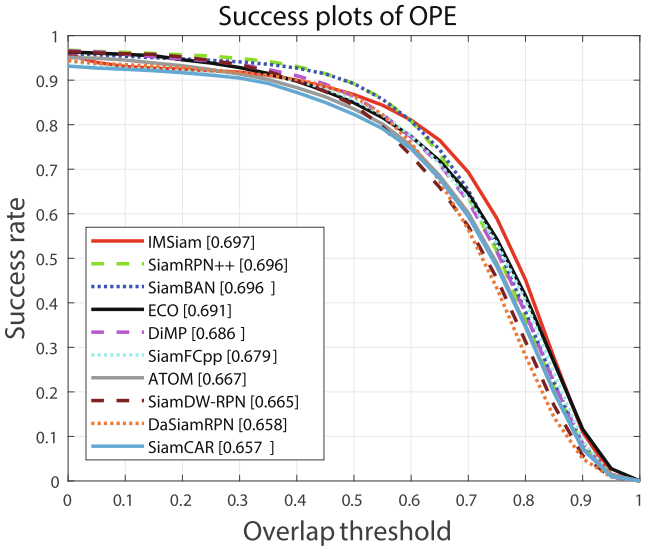
We use the ResNet50 with atrous convolution as the Siamese backbone network for all experiments, which is initialized with the pre-trained parameters on the ImageNet classification task provided in [13]. All the weights and biases of the newly added convolution layers are initialized with the Kaiming algorithm [53]. The training splits of COCO2017 [54], ImageNet VID2015 [55], LaSOT [20], and GOT10k [21] datasets are used to train our model. We extract pairs of the template and search images containing the same target as positive training samples and extract pairs containing different targets as negative training samples with a probability of 0.2. Several effective data augmentations are applied including random stretch and translation, blur, color jitter and graying. The whole model is trained with the multi-task loss by Eq. (15). We use synchronized SGD over the 4 GPUs with a total of 64 image pairs per minibatch (16 pairs per GPU) and train 320,000 image pairs per epoch. Weight decay and momentum are set as 0.0001 and 0.9. The stem and stage-0 of the ResNet50 backbone are frozen and the rest of the network is trained for 20 epochs in total. A warm-up scheduler training strategy is adopted where the learning rate increases linearly from $1e^{-3}$ to $5e^{-3}$ for the first five epochs and then decays exponentially by 0.8. Moreover, we fine-tune the backbone network with one-tenth of the current learning rate.

4.1.2. Evaluation details

We use a single RTX2080Ti GPU for tracking evaluation. Note that our tracking method only has two hyper-parameters including the weight of cosine window and the linear interpolation factor of target size. we perform comprehensive comparisons with the state-of-the-art on the OTB dataset, including OTB2013 [18] and OTB2015 [3]. A detailed ablation analysis of our approach on the OTB2015 dataset is also provided. Last, we report our main results on the other four datasets including VOT2018 [19], VOT2019 [4], GOT10k [21] and LaSOT [20]. The average tracking speed is about 52 FPS without optimization.



(a) OTB2013



(b) OTB2015

Fig. 6. Success plots of OPE on OTB2013 [18] and OTB2015 [3] dataset.

4.2. OTB benchmark

OTB benchmark consists of two subsets, i.e., OTB2013 [18] and OTB2015 [3]. Specifically, OTB2013 contains 51 fully annotated sequences with 11 different attributes including scale variation, illumination variation, fast motion, background clutter, motion blur, in-plane rotation, out-of-plane rotation, deformation, low resolution, out of view, and occlusion. Later, OTB2015 extends OTB2013 to 100 videos. We conduct a quantitative analysis on OTB by the success plot in the one-pass evaluation (OPE) protocol. Specifically, the tracking result in a frame is declared successful if its overlap ratio exceeds a pre-defined threshold. The success plot shows the percentage of successful frames at the overlap threshold varies from 0 to 1 and we use the Area Under the Curve (AUC) metric is used to assess a tracker.

4.2.1. State-of-the-art comparision

We compare our proposed approach IMSiam with the recent state-of-the-art methods including SiamBAN [6], SiamFC++ [7], SiamCAR [5], SiamRPN++ [13], SiamDW-RPN [14], DaSiamRPN [12], DiMP [39], ATOM [38] and ECO [29]. As is shown in Fig. 6, our approach achieves the best AUC score both on OTB2013 [18] and OTB2015 [3]. On the OTB2013 dataset, our approach achieves an AUC score of 0.711, which outperforms the previous best tracker ECO [29]. Moreover, our tracker can run over 50 FPS while the ECO tracker only runs at less than 10 FPS. On the OTB2015 dataset, our approach achieves an AUC score of 0.697, which outperforms the best anchor-based tracker SiamRPN++ [13] and the best anchor-free tracker SiamBAN [6] while employing the same backbone network.

4.2.2. Ablation analysis

The main contribution of this work is that we propose a novel anchor-free Siamese network for visual tracking which contains the proposed matching-adaptive neck network and the IoU-aware head network. To verify the effectiveness of the above two modules, we conduct an ablation analysis on the OTB2015 dataset. We remove the L1 difference and search identity feature maps from the neck as well as the IoU prediction head as the baseline model. It can be found from Table 1 that the baseline method achieves an AUC score of 0.660. By adding the L1 difference and search identity feature maps to the neck network, the AUC score improves to 0.665 and 0.680 respectively. After we add an IoU-aware head to the baseline, the performance is significantly improved to 0.679, suggesting that our proposed IoU head can help the classification results to obtain more accurate target bounding boxes. By adding the L1 difference and search identity feature maps to the neck network, the AUC score improves to 0.684 and 0.697 respectively. This suggests that fusing multi-type feature maps is critical to the tracking results.

Here we further analyze the matching feature sampling layer to help understand the contributions of the L1 difference and identity

Table 1
Ablation study of our proposed tracker on the OTB2015 dataset [3].

Subnetwork	Backbone	Neck			Head		AUC↑
	Stage 1,2,3	PDW-XCorr	L1 Difference	Search Identity	Cls + Cen + Reg	IoU	
Baseline	✓	✓			✓		0.660
IMSiam	✓	✓	✓		✓		0.665
	✓	✓	✓	✓	✓		0.680
	✓	✓			✓	✓	0.679
	✓	✓	✓		✓	✓	0.684
	✓	✓	✓	✓	✓	✓	0.697

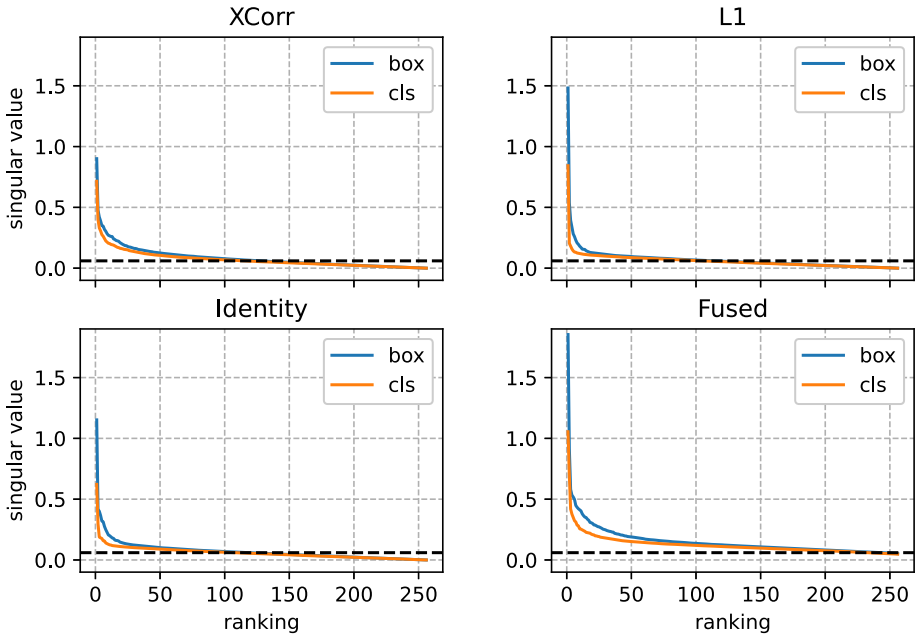
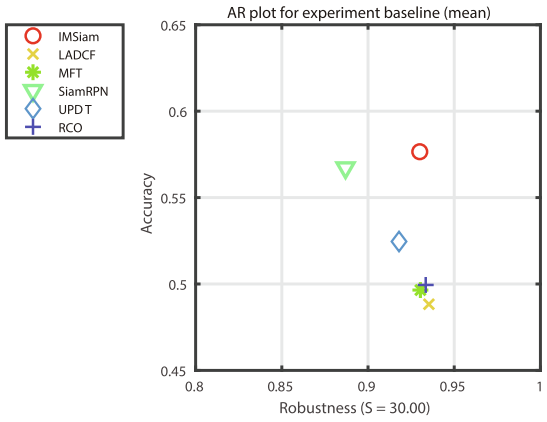
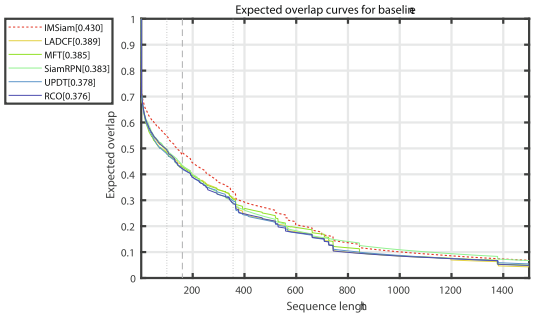


Fig. 7. Singular value distribution of XCorr, L1, Identity and fused projection matrix.



(a) Accuracy-Robustness rank plot for 6 trackers on VOT2018 [19]. Better trackers are closer to the top right corner.



(b) Expected Average Overlap (EAO) curve for 6 trackers on VOT2018 [19].Our tracker has much better performance than the compared trackers.

Fig. 8. Comparision with the top-5 trackers in VOT2018 challenge [19].

maps. In our implementation, we have the L1 difference maps $f_1 \in \mathbb{R}^{256\times29\times29}$, the search identity maps $f_2 \in \mathbb{R}^{256\times29\times29}$ and the PDW-XCorr maps $f_3 \in \mathbb{R}^{256\times29\times29}$. As in Section 3.2, we firstly concatenate the above three types of feature maps, which outputs the fused maps $F \in \mathbb{R}^{768\times29\times29}$. Then a 1×1 convolution layer is used to sample the matching features to perform the subsequent task, where the projection matrix $W \in \mathbb{R}^{256\times768}$ is used to compute:

$$\Phi = WF. \tag{20}$$

The Eq. (20) can be reformulated as:

$$\Phi = [L \quad I \quad X]\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, = Lf_1 + If_2 + Xf_3, \tag{21}$$

where $W = [L\ I\ X], L, I, X \in \mathbb{R}^{256\times256}$ and they are responsible for extracting matching information from the L1 difference f_1, search identity f_2 and PDW-Xcorr maps f_3, respectively. Fig. 7 shows the singular value distribution of X, L, I and W matrixes for classification and box regression tasks, respectively. As shown in Fig. 7, the matrix X, L, and I matrixes all have relatively fewer large singular values than the fused W matrix, which indicates that they all sample a small amount of valid matching information and complement to each other. Moreover, there is a significant difference in the singular value distribution between the classification and box regression tasks, indicating that the features Φ_{cls} and Φ_{reg} for the above two tasks are also very different.

4.3. VOT benchmark experiments

For completeness, we also present the evaluation results on the VOT benchmark [19,4]. Note that the VOT challenge protocol is different from the protocol in the OTB benchmark. Specifically, a tracker is automatically re-initialized five frames after the failure

Table 2
Ranked results for the recent proposed trackers on VOT2018 [19]. **First**, *second* and third entries for accuracy(A), robustness(R), and EAO are reported.

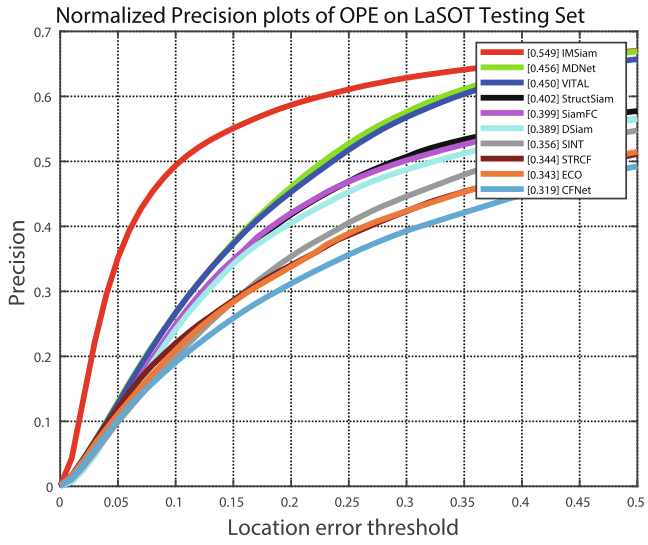
Trackers	SiamRPN++ [13]	SiamFC++ [7]	ATOM [38]	SiamBAN [6]	DiMP [39]	IMSiam
A ↑	**0.600**	0.587	0.590	0.597	0.597	*0.599*
R ↓	0.234	0.183	0.204	*0.178*	**0.153**	0.206
EAO ↑	0.417	0.426	0.401	**0.452**	*0.440*	0.430

Table 3
Ranked results for the recently proposed trackers on VOT2019 [4]. **First**, *second* and third entries for accuracy(A), robustness(R), and EAO are reported. Note that the SiamBAN tracker reports its tracking results in VOT2019 real-time experiments, which is obtained under stricter conditions.

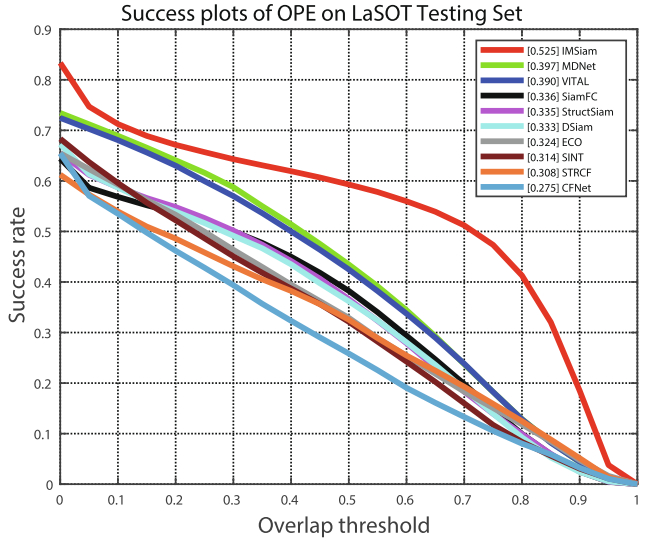
Trackers	SiamRPN++ [13]	SiamMask [56]	SPM [16]	SiamBAN* [6]	DiMP [39]	IMSiam
A↑	0.599	0.594	0.577	*0.602*	0.594	**0.604**
R↓	0.482	0.461	0.507	*0.397*	**0.278**	0.446
EAO ↑	0.285	0.287	0.275	*0.327*	**0.379**	0.312

Table 4
Comparisons on the GOT10k dataset [21].

Method	AO ↑	$SR_{0.5}$↑	$SR_{0.75}$↑	Hz	Hardware	Language
ECO	0.316	0.309	0.111	2.62fps	CPU	Matlab
SiamRPN	0.483	0.581	0.270	97.55fps	Titan X	Python
SiamRPN++	0.517	0.616	0.359	49.83fps	RTX 2080Ti	Python
SPM	0.513	0.593	0.359	72.30fps	Titan Xp	Python
SiamCAR	0.569	0.670	0.415	52.27fps	RTX 2080Ti	Python
SiamFC++	0.595	0.695	0.479	90fps	RTX 2080Ti	Python
ATOM	0.556	0.634	0.402	30fps	GTX1080	Python
DiMP18	0.579	0.672	0.446	57fps	GTX1080	Python
DiMP50	0.611	0.717	0.492	34.05fps	GTX1050	Python
IMSiam	0.601	0.698	0.491	52.80fps	RTX 2080Ti	Python



(a)



(b)

Fig. 9. Evaluation results of trackers on LaSOT testing dataset. We compare our tracker with the top-9 methods reported in [20].

and the performance is measured in terms of accuracy, robustness and expected average overlap (EAO). The accuracy of a tracker on a sequence is calculated by the average per-frame IoU between its predicted bounding box and the ground-truth. The robustness score evaluates the average number of tracking failures. And the EAO score measures the average no-rest overlap of a tracker on several short-term sequences. Here we conduct experiments on the recent VOT datasets including VOT2018 [19] and VOT2019 [4].

4.3.1. VOT2018 [19]

We evaluate our tracker on the VOT2018 [19] dataset, which consists of 60 fully-annotated challenging sequences. And our tracker achieves an EAO score of 0.430. We first compare our tracker with the top 5 trackers reported in [19]. As is shown in Fig. 8, our IMSiam tracker is ranked top overall. We also report the comparison results with the recent methods including SiamBAN [6], SiamFC++ [7], SiamRPN++ [13], DiMP [39], and ATOM [38]. Table 2 shows that our tracker is competitive with the state-of-the-art. Specifically, our tracker outperforms the best anchor-based Siamese tracker SiamRPN++ by 3.1% in terms of EAO score. DiMP achieves a lower failure rate through online updates, but our accuracy is higher than it. And our tracker also achieves a higher accuracy score than SiamBAN, which shows that our method can accurately estimate the target bounding box.

4.3.2. VOT2019 [4]

We further evaluate our tracker on the VOT2019 [4] dataset. The VOT2019 sequences were replaced by 20% compared to the VOT2018 and more challenging. And our IMSiam tracker is compared with five recent state-of-the-art methods including SiamRPN++ [13], SiamMask [56], SPM [16], ATOM [38] and DiMP [39]. Table 3 shows the tracking results in terms of Accuracy(A), Robustness(R) and EAO. Our tracker achieves state-of-the-art performance. Compared to SiamBAN and DiMP, our tracker achieves a slightly higher failure rate. However, our tracker achieves the best accuracy score among all the compared trackers. This shows our method can estimate the target bounding box accurately by the proposed IoU head. Moreover, there are only two hyperparameters during tracking, making our tracker simpler than other compared trackers.

4.4. Experiments on large-scale tracking datasets

4.4.1. GOT10k [21]

GOT10k [21] is a large-scale tracking dataset containing over 10,000 videos, 180 of which form the test set used for evaluation. The average overlap (AO) calculated by the AUC of success plot and the success rate (SR) with different thresholds are used to evaluate the tracker. We upload the tracking results to the official website and the analysis is reported in Table 4. We compare our tracker with state-of-the-art methods including ECO [29], SiamRPN [9], SiamRPN++ [13], SPM [16], SiamCAR [5], SiamFC++ [7], ATOM [38] and DiMP [39]. Our IMSiam tracker achieves an AO score of 0.601 and ranks the second, verifying the strong generalization abilities. Compared to the top-ranked DiMP with ResNet50, our tracker runs faster in the tracking initialization phase because we need not draw samples to train an online classifier. Moreover, the tracking results of our method are deterministic with fewer hyperparameters, while ATOM and DiMP are random and have much more hyperparameters.

4.4.2. LaSOT [20]

LaSOT [20] is a recent released benchmark for single object tracking, which contains 1400 sequences. The normalized precision plots and success plots in one-pass evaluation are used to assess the trackers. We compare our IMSiam tracker with the top

Table 5
Comparison with the recently proposed trackers on LaSOT [20] testing dataset.

Trackers	C-RPN [15]	SiamBAN [6]	SiamRPNpp [13]	SiamCAR [5]	ATOM [38]	DiMP [39]	IMSiam
AUC ↑	0.455	0.514	0.496	0.507	0.515	**0.568**	*0.525*

9 trackers reported in [20]. As is shown in Fig. 9, our tracker achieves the best performance both in precision plots and success plots. And we also compare with the recent methods including C-RPN [15], SiamRPN++ [13], SiamCAR [5], SiamBAN [6], ATOM [38] and DiMP [39], in terms of the AUC score in Table 5. Our tracker ranks second and significantly improves the anchor-free Siamese trackers such as SiamCAR [5] and SiamBAN [6]. And our tracker also outperforms the best anchor-based Siamese tracker SiamRPN ++ [13] by 5.8%.

5. Conclusion

In this paper, we present a novel IoU-aware matching-adaptive Siamese network for visual tracking. Specifically, we introduce a matching-adaptive network which fuses multi-type feature maps and adaptively draws matching information for classification and bounding box regression. We also introduce an IoU-aware head to assess the quality of regressed bounding boxes, which can predict the IoU and tune the corresponding classification score. By the above design, the whole model is fully anchor-free and can be trained end-to-end while it only has two hyper-parameters during tracking. Experiments show that our method obtains state-of-the-art results, with real-time running speed.

CRediT authorship contribution statement

Ke Tan: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **Ting-Bing Xu:** Validation, Writing - review & editing. **Zhenzhong Wei:** Resources, Supervision, Project administration, Funding acquisition.

Declaration of Competing 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.

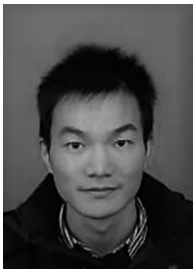
Acknowledgment

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 52127809, 51625501.

References

[1] A.W. Smeulders, D.M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah, Visual tracking: an experimental survey, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2013) 1442–1468.
[2] J. Shen, D. Yu, L. Deng, X. Dong, Fast online tracking with detection refinement, IEEE Trans. Intell. Transp. Syst. 19 (1) (2017) 162–173.
[3] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1834–1848, https://doi.org/10.1109/TPAMI.2014.2388226.
[4] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg, et al., The seventh visual object tracking vot2019 challenge results, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019, pp. 2206–2241.
[5] D. Guo, J. Wang, Y. Cui, Z. Wang, S. Chen, Siamcar: Siamese fully convolutional classification and regression for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 6269–6277.
[6] Z. Chen, B. Zhong, G. Li, S. Zhang, R. Ji, Siamese box adaptive network for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 6668–6677.
[7] Y. Xu, Z. Wang, Z. Li, Y. Yuan, G. Yu, Siamfc++: towards robust and accurate visual tracking with target estimation guidelines., in: AAAI, 2020, pp. 12549–12556.
[8] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, P.H. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 850–865.
[9] B. Li, J. Yan, W. Wu, Z. Zhu, 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] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[11] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, et al., The visual object tracking vot2017 challenge results, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 1949–1972.
[12] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 101–117.
[13] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, 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.
[14] Z. Zhang, 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.
[15] H. Fan, H. Ling, Siamese cascaded region proposal networks for real-time visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7952–7961.
[16] G. Wang, C. Luo, Z. Xiong, W. Zeng, Spm-tracker: series-parallel matching for real-time visual object tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3643–3652.
[17] Z. Tian, C. Shen, H. Chen, T. He, Fcos: fully convolutional one-stage object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9627–9636.
[18] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: a benchmark, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2411–2418.
[19] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, et al., The sixth visual object tracking vot2018 challenge results, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 0–0.
[20] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, 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.
[21] L. Huang, X. Zhao, K. Huang, Got-10k: a large high-diversity benchmark for generic object tracking in the wild, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[22] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu, H. Huang, Occlusion-aware real-time object tracking, IEEE Trans. Multimedia 19 (4) (2016) 763–771.
[23] S. Hong, T. You, S. Kwak, B. Han, Online tracking by learning discriminative saliency map with convolutional neural network, in: International Conference on Machine Learning, 2015, pp. 597–606.
[24] M. Danelljan, G. Hager, F. Shahbaz Khan, M. Felsberg, Convolutional features for correlation filter based visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2015, pp. 58–66.
[25] C. Ma, J.-B. Huang, X. Yang, M.-H. Yang, Hierarchical convolutional features for visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3074–3082.
[26] D.S. Bolme, J.R. Beveridge, B.A. Draper, Y.M. Lui, Visual object tracking using adaptive correlation filters, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2544–2550.
[27] J.F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Trans. Pattern Anal. Mach. Intell. 37 (3) (2014) 583–596.
[28] M. Danelljan, A. Robinson, F.S. Khan, M. Felsberg, Beyond correlation filters: learning continuous convolution operators for visual tracking, in: European Conference on Computer Vision, Springer, 2016, pp. 472–488.
[29] M. Danelljan, G. Bhat, F. Shahbaz Khan, M. Felsberg, Eco: efficient convolution operators for tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6638–6646.
[30] F. Li, C. Tian, W. Zuo, L. Zhang, M.-H. Yang, Learning spatial-temporal regularized correlation filters for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4904–4913.
[31] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, M. Felsberg, Unveiling the power of deep tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 483–498.
[32] H. Li, Y. Li, F. Porikli, Deeptrack: learning discriminative feature representations online for robust visual tracking, IEEE Trans. Image Process. 25 (4) (2015) 1834–1848.
[33] H. Nam, 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.
[34] H. Nam, M. Baek, B. Han, Modeling and propagating cnns in a tree structure for visual tracking, arXiv preprint arXiv:1608.07242 (2016).
[35] Y. Song, C. Ma, L. Gong, J. Zhang, R.W. Lau, M.-H. Yang, Crest: convolutional residual learning for visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2555–2564.

[36] H. Hu, B. Ma, J. Shen, H. Sun, L. Shao, F. Porikli, Robust object tracking using manifold regularized convolutional neural networks, IEEE Trans. Multimedia 21 (2) (2018) 510–521.
[37] X. Lu, C. Ma, J. Shen, X. Yang, I. Reid, M.-H. Yang, Deep object tracking with shrinkage loss, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[38] M. Danelljan, G. Bhat, F.S. Khan, M. Felsberg, Atom: accurate tracking by overlap maximization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4660–4669.
[39] G. Bhat, M. Danelljan, L.V. Gool, R. Timofte, Learning discriminative model prediction for tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6182–6191.
[40] M. Danelljan, L.V. Gool, R. Timofte, Probabilistic regression for visual tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7183–7192.
[41] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P.H. Torr, End-to-end representation learning for correlation filter based tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2805–2813.
[42] X. Dong, J. Shen, Triplet loss in siamese network for object tracking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 459–474.
[43] X. Dong, J. Shen, D. Wu, K. Guo, X. Jin, F. Porikli, Quadruplet network with one-shot learning for fast visual object tracking, IEEE Trans. Image Process. 28 (7) (2019) 3516–3527.
[44] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, F. Porikli, Hyperparameter optimization for tracking with continuous deep q-learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 518–527.
[45] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, F. Porikli, Dynamical hyperparameter optimization via deep reinforcement learning in tracking, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[46] A. He, C. Luo, X. Tian, W. Zeng, A twofold siamese network for real-time object tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4834–4843.
[47] Z. Liang, J. Shen, Local semantic siamese networks for fast tracking, IEEE Trans. Image Process. 29 (2019) 3351–3364.
[48] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, S. Maybank, Learning attentions: residual attentional siamese network for high performance online visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4854–4863.
[49] J. Shen, X. Tang, X. Dong, L. Shao, Visual object tracking by hierarchical attention siamese network, IEEE Trans. Cybern. 50 (7) (2019) 3068–3080.
[50] X. Dong, J. Shen, L. Shao, F. Porikli, Clnet: a compact latent network for fast adjusting siamese trackers, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16, Springer, 2020, pp. 378–395.
[51] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[52] B. Jiang, R. Luo, J. Mao, T. Xiao, Y. Jiang, Acquisition of localization confidence for accurate object detection, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 784–799.
[53] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[55] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[56] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, 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.



Ke Tan received the B.Eng. from the Department of Instrument Science and Opto-electronics Engineering of Beihang University, Beijing, China, in 2014. He is now a Ph.D. student in the Department of Instrument Science and Opto-electronics Engineering of Beihang University, Beijing, China. His current research interests are object tracking, machine learning and image processing.

Tingbing Xu received the BS degree in automation from China University of Petroleum, Qingdao, China, in 2014 and the PhD degree in computer applied technology from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences, Beijing, China, in 2020. He is currently a postdoctoral fellow at the School of Instrumentation Science and Opto-electronics Engineering, Beihang University, Beijing, China. He was a visiting researcher at the Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka, Japan, in 2018. His research interests include deep learning, machine learning, pattern recognition, handwriting recognition, and computer vision.

Zhenzhong Wei received his B.S. degree from the Automation Department of Beijing Institute of Petro-Chemical Technology, China, in 1997, and his M.S. and Ph.D. degree from the School of Automation Science and Electrical Engineering at Beihang University, China, in 1999 and 2003, respectively. He was awarded as special professor of Yangtze River Scholar in 2016. He is now a professor in the School of Instrumentation Science and Opto-electronics Engineering, Beihang University, China. His research interests include machine vision and artificial intelligence.