

# SiamCorners: Siamese Corner Networks for Visual Tracking

Kai Yang, Zhenyu He\*, *Senior Member, IEEE*, Wenjie Pei\*, Zikun Zhou, Xin Li, Di Yuan and Haijun Zhang

**Abstract**—The current Siamese network based on region proposal network (RPN) has attracted great attention in visual tracking due to its excellent accuracy and high efficiency. However, the design of the RPN involves the selection of the number, scale, and aspect ratios of anchor boxes, which will affect the applicability and convenience of the model. Furthermore, these anchor boxes require complicated calculations, such as calculating their intersection-over-union (IoU) with ground truth bounding boxes. Due to the problems related to anchor boxes, we propose a simple yet effective anchor-free tracker (named Siamese corner networks, SiamCorners), which is end-to-end trained offline on large-scale image pairs. Specifically, we introduce a modified corner pooling layer to convert the bounding box estimate of the target into a pair of corner predictions (the bottom-right and the top-left corners). By tracking a target as a pair of corners, we avoid the need to design the anchor boxes. This will make the entire tracking algorithm more flexible and simple than anchor-based trackers. In our network design, we further introduce a layer-wise feature aggregation strategy that enables the corner pooling module to predict multiple corners for a tracking target in deep networks. We then introduce a new penalty term that is used to select an optimal tracking box in these candidate corners. Finally, SiamCorners achieves experimental results that are comparable to the state-of-art tracker while maintaining a high running speed. In particular, SiamCorners achieves a 53.7% AUC on NFS30 and a 61.4% AUC on UAV123, while still running at 42 frames per second (FPS).

**Index Terms**—Visual tracking, Siamese network, top-left corner, bottom-right corner.

## I. INTRODUCTION

VISUAL trackers based on Siamese networks [1], [2], [3], [4] have achieved state-of-the-art results on various tracking benchmark datasets [5], [6], [7], [8], [9]. A key component of state-of-the-art trackers [1], [10], [11] is based on anchor boxes, which are boxes of different sizes and aspect ratios that select as tracking candidates. Anchor boxes are widely used in visual tracking [1], [3], [10], [11], and can achieve a good trade-off in speed-accuracy. These methods [1], [3], [10], [11] first generate anchor boxes densely over the image, they then find the candidate regions of the target by learning a strong classifier, and they finally refine their coordinates through a regression network.

K. Yang, W. Pei, Z. Zhou, D. Yuan and H. Zhang are with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, 518055, China.

X. Li is with Peng Cheng Laboratory, Shenzhen, China.

Z. He is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, 518055, China, and also with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, China, e-mail: (zhenyuhe@hit.edu.cn).

\*Zhenyu He and Wenjie Pei are corresponding authors.

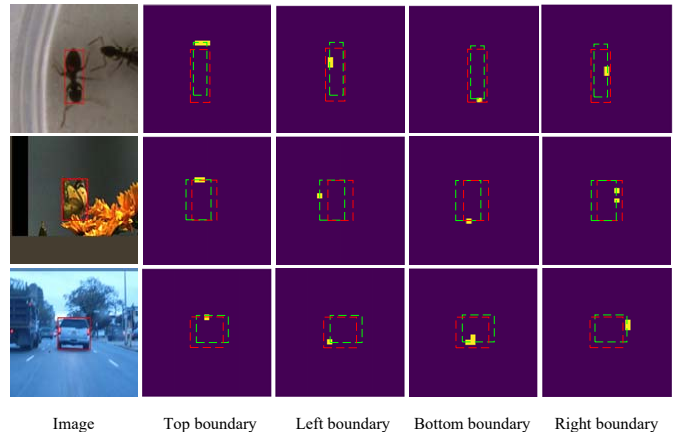


Fig. 1. Visualization results of feature maps (keep only the highest score for clarity) of the four boundaries (green box) predicted by Siamese network. The feature maps of the four boundaries are fed to a modified corner pooling layer to predict the corner heatmaps and offsets of the green box. Specifically, we combine the offset prediction and penalty strategies to make the green box close to the ground truth box (red box).

However, trackers based on anchor boxes have two drawbacks. First, these trackers need very large-scale anchor boxes, e.g. more than 2k in SiamRPN [11] and more than 3k in SiamRPN++ [1]. It is because that the anchor-based tracker is trained to classify whether each anchor box contains a target region, and a large number of anchor boxes are required to ensure that they have sufficient overlapping regions with the ground truth boxes. In training, the positive and negative samples are defined by calculating the IoU between anchor boxes and ground truth boxes [12]. They [1], [3], [10], [11] simply think that the anchor boxes are positive samples if the IoU is larger than a hand-crafted threshold and are negative samples if the IoU is less than a hand-crafted threshold. However, recent research [13] indicates that this fixed hand-craft threshold will affect the performance of anchor-based algorithms. As a result, a tiny part of the anchor boxes can ensure overlap with ground truth boxes. In this design, this will produce a huge imbalance between positive samples and negative samples [1] and slows down training [14], [15]. Furthermore, the imbalance of positive and negative samples will have a huge impact on a model's performance [13], [16]. For example, the training for the classification model is inefficient as most locations are easy negative samples, which will lead to a degraded classification model [15]. If the classifier cannot produce a correct candidate region for the target location, then the regression network will not be

accurate [17], [18].

Second, trackers with hand-crafted anchor boxes will introduce many design options. This includes how to select the number, size, and aspect ratio of anchor boxes. This design is mostly inspired by ad-hoc heuristics and it can become more complex when a network predicts multiple-level feature maps, each of which uses different features and has specific anchor boxes [1], [14], [15], [19]. These parameters in anchor-based trackers need to be tuned carefully. Even with careful design, it is difficult for trackers to deal with large changes of target shapes, especially for small targets, due to the fixed scales and aspect ratios of the anchor boxes.

In contrast, the location and boundary of the target that humans can identify are not obtained by predefined candidate boxes [4]. Therefore, we can design a tracking algorithm without any predefined anchor boxes. Motivated by these analyses, we introduce a SiamCorners method for visual tracking, which makes our tracking algorithm more flexible and general by discarding anchor boxes. Our proposed method consists of four steps. First, we crop four boundary images containing target boundary information from an initial frame as the template images of the SiamCorners. Specifically, the four boundary images regard the top, left, bottom and right boundary of the bounding box of the initial frame as the initial locations, respectively, and it then crops at a specific ratio of the width or height of the box. The SiamCorners then predicts the feature maps (see figure 1) of the four boundaries of the bounding box based on the input of template images and search images. Second, inspired by the recently proposed CornerNet [14], [20], which is used to predict a bounding box as a pair of corners in object detection. However, the original CornerNet is class-specific, and hence cannot be used directly for visual tracking. Consequently, we introduce a modified corner pooling layer to help the network integrate the target-specific feature in the Siamese framework. The entire corner pooling layer is used to predict the multiple top-left corners and bottom-right corners of the bounding box by the predicted feature maps of the four boundaries. Third, a decoding process is performed to predict the corner locations based on the heatmaps and offset maps. It is worth noting that we chose a Non-maximum-suppression (NMS) operation to remove some abnormal corners. Finally, multi-level features fusion and some penalty strategies are adopted to select the best tracking bounding box. Intuitively, features from earlier convolution outputs of deep networks will contain rich low-level feature information such as shape, color, which is important for tracking localization. Features from the latter convolution outputs of the deep networks will contain rich semantic information that is beneficial to deal with some challenging factors like huge deformation and background clutter. In our design, multi-layer features are fed into the multiple SiamCorners modules individually. We then use a new penalty function to select the best tracking box from the candidate corners generated by the layer-wise aggregation strategy. Specifically, we first employ the location and scale penalty function that considers the changes of width, height, and center of the tracking target compared to the previous frame. This penalty strategy is used to re-rank the corners' score to remove the outliers. We then utilize a penalty term to

suppress large changes in the ratio and size of the corners. Subsequently, we select the corners with the largest score as a candidate tracking box. Finally, a linear interpolation is employed to make the tracking box changing smoothly. This design simplifies the output of the whole network and eliminates the complex design of the anchor boxes.

In summary, the proposed SiamCorners mainly includes the following contributions:

- We propose an anchor-free proposal network (SiamCorners) that avoids many hand-crafted configurations during designing the anchor boxes and makes our method more flexible and general.
- We present a modified corner pooling layer to integrate target-specific information into the corner prediction.
- We formulate a multi-layer features fusion strategy to help SiamCorners benefit from both low-level information and high-level information.
- We introduce a novel penalty term that considers the changes of the center, width, and height of the target in the adjacent frames, which makes SiamCorners more suitable for tracking tasks.

Based on these contributions, we demonstrate the effectiveness and efficiency of SiamCorners on five tracking benchmark datasets: OTB100 [5], UAV123 [7], LaSOT [6], NFS30 [8] and TrackingNet [9]. Our SiamCorners approach achieves state-of-the-art results on the five datasets. Moreover, the wide range of ablation experiments are performed on the UAV123 and the LaSOT datasets.

## II. RELATED WORKS

Generic object tracking has made great progress in recent years following the development of several new methods [1], [2], [3], [4], [21], [22], [23], [24]. In particular, methods based on Siamese networks play an important role in visual tracking.

**Siamese Networks:** Recently, trackers based on Siamese networks have received much attention because of their well-balanced in accuracy-efficiency. SiameseFC [25] calculates the similarity prediction of template images and search images through a full-convolutional network, while running over 100FPS. CFNet [26] introduces correlation filter layers into the Siamese tracking framework, which can achieve end-to-end representation learning of tracking targets. However, its performance is not significantly improved compared to SiameseFC. SiamRPN [11] develops the tracking task as a one-shot detection task by introducing region proposal subnetwork in the Siamese network, which is trained offline in an end-to-end manner on large-scale image pairs. The region proposal network contains two branches, one for classification branch and another for regression branch. The classification branch predicts the target location based on the fixed anchor boxes, while the regression branch fine-tunes the predicted target coordinates. Therefore, the design of anchor boxes has a great impact on the final tracking results. DaSiamRPN [10] further improves the discriminative ability of the model by introducing an effective sampling strategy to control the imbalanced distribution of the training data. However, note that these trackers [25], [26], [11], [10] built their algorithm in shallow networks

similar to AlexNet [27], the components they proposed did not benefit from low-level and high-level feature information due to relatively shallow networks. While our method uses multiple SiamCorners modules to predict multiple candidate corners according to the layer-wise aggregation features of the deep networks. Similarly, SiamRPN++ [1] enables SiamRPN tracker to take advantage of deep features by introducing deeper networks, such as ResNet-50 [28] or deeper networks. Unlike anchor-based trackers [1], [10], [11], our SiamCorners tracker is more flexible and general since it eliminates the design of anchor boxes.

**Anchor-free Mechanism:** Anchor-free trackers recently became popular in visual tracking tasks due to their relatively simple and superior performance. FCAF [29] and Ocean [18] respectively introduce a simple anchor-free tracker by using a variant of the FCOS [12] detector. They [18], [29] first generate some hand-crafted points instead of a large number of anchor boxes as training samples, and then use a classification network to find some points that may be the locations of a tracking target. Finally, they utilize a regression network to predict a 4D vector, which represents the distance from the target location to the four sides of the bounding box. Unlike these anchor-free trackers, our SiamCorners directly predicts the target box as a pair of corners, which does not include a classification network to determine the coarse location in some hand-crafted points. Therefore, SiamCorners maintains a high-efficiency running speed by eliminating the classification phase of the anchor-free trackers [18], [29].

Recently, corner-based detectors have attracted widespread attention in the object detection community due to their relatively new paradigms. CornerNet [14] was the first to propose a corner-based detector that converts the bounding box of the target into a pair of corner predictions. CornerNet uses an hourglass network [30] to obtain the heatmaps including the top-left corners and the bottom-right corners of the bounding boxes. It then uses a series of feature embeddings to match these corners. To accurately predict the target, CornerNet introduces a corner pooling to localize the corner. CornerNet-Lite [20] utilizes CornerNet as a baseline to introduce an attention mechanism so that the detector does not require detailed processing for each pixel in the input image, thereby improving the inference efficiency of the entire detector. CenterNet [31] regards each detection target as a triplet, which predicts the centers of the bounding boxes along with corners. MartixNets [32] proposes an aspect ratio and scale aware architecture so that CornerNet can eliminate corner pooling operation. Instead of matching corners by feature embeddings in CornerNet, CentripetalNet [33] matches the predicted corners by whether the centripetal shift results of the corners are aligned, where the centripetal shift denotes the spatial shift from the corner locations to the center of the bounding box.

There are two key differences between tracking and detection tasks. First, anchor-free detectors usually aim to detect different categories of targets in an input image, while in tracking, we aim to match specific targets in search images. Therefore, we follow a common Siamese architecture [1] that can discriminate the target from the search images. Second,

the detector based on anchor-free usually assumes that the target category is predefined. The common dataset MS COCO [34] of object detection only contains 91 target categories. Thus, independent corner pooling layers are trained in [14], [20], [31], [32], [33] for each object class. In contrast, the target class is unknown in tracking [1], [35], [36]. Furthermore, unlike the object detection tasks, the target does not belong to any pre-defined classes or is represented by existing training datasets. Instead, target-specific corner pooling layers are required for test frames incorporating target information in Siamese framework.

### III. PROPOSED METHOD

#### A. Overview

In this work, we propose an anchor-free method, SiamCorners, which detects a target as a pair of corners. A common Siamese network is used to integrate target-specific information into the tracking architecture.

Fig. 2 provides an overview of SiamCorners. As illustrated in Fig. 2, the template branch has four inputs: i) the top-boundary image  $z_t$ , ii) the left-boundary image  $z_l$ , iii) the bottom-boundary image  $z_b$ , iv) the right-boundary image  $z_r$  (see Fig. 3). The target images of the top (left) boundary and the bottom (right) boundary respectively take the four boundaries of the target’s bounding box as the initial location, and are cropped from  $t_{w,h}$  (we set  $t_{w,h} = 0.5$  in all experiments) times the width (height) of the bounding box of the original image, respectively. Furthermore, the other regions of the boundary images are constant padding. Then, we use the same data augmentation strategy as SiamRPN++ [1] to resize the four boundary images to a fixed size. Afterwards, the four boundary images of the bounding box are fed to ResNet-50 [28] network, respectively. The feature maps from different layers, namely *conv3*, *conv4*, and *conv5* blocks of ResNet-50, are fed to the two branches. Specifically, the feature maps of the top and the left boundary images are fed to the top-left corner prediction branch, and the feature maps of the bottom and the right boundary images are fed to the bottom-right corner prediction branch. The features of different layers in ResNet-50 are followed by a convolution network. Depth-wise correlation operations are further performed to obtain correlation features. The correlation features are then fed to the corner pooling layers to get the corner heatmaps and offsets, respectively (see Section III-C). After obtaining the predicted heatmaps and offsets, a simple decoding process is used to obtain the corner candidates (see Section III-D). Finally, the multi-layer feature fusion and the post-processing operation are used to get the final bounding box (see Section III-E).

#### B. Tracking Corners Loss

We predict two sets of corner heatmaps that are used to calculate the top-left corners and the bottom-right corners of the target, respectively. Let  $\tilde{x} \in \mathbb{R}^{H \times W}$  be a search image, where  $H$  and  $W$  are the height and width of the image, respectively. The corner heatmaps of the convolutional network are denoted as  $\hat{X} \in [0, 1]^{\frac{H}{s} \times \frac{W}{s}}$ , where  $s$  is a scale factor mapped from the search image to the corner heatmaps.

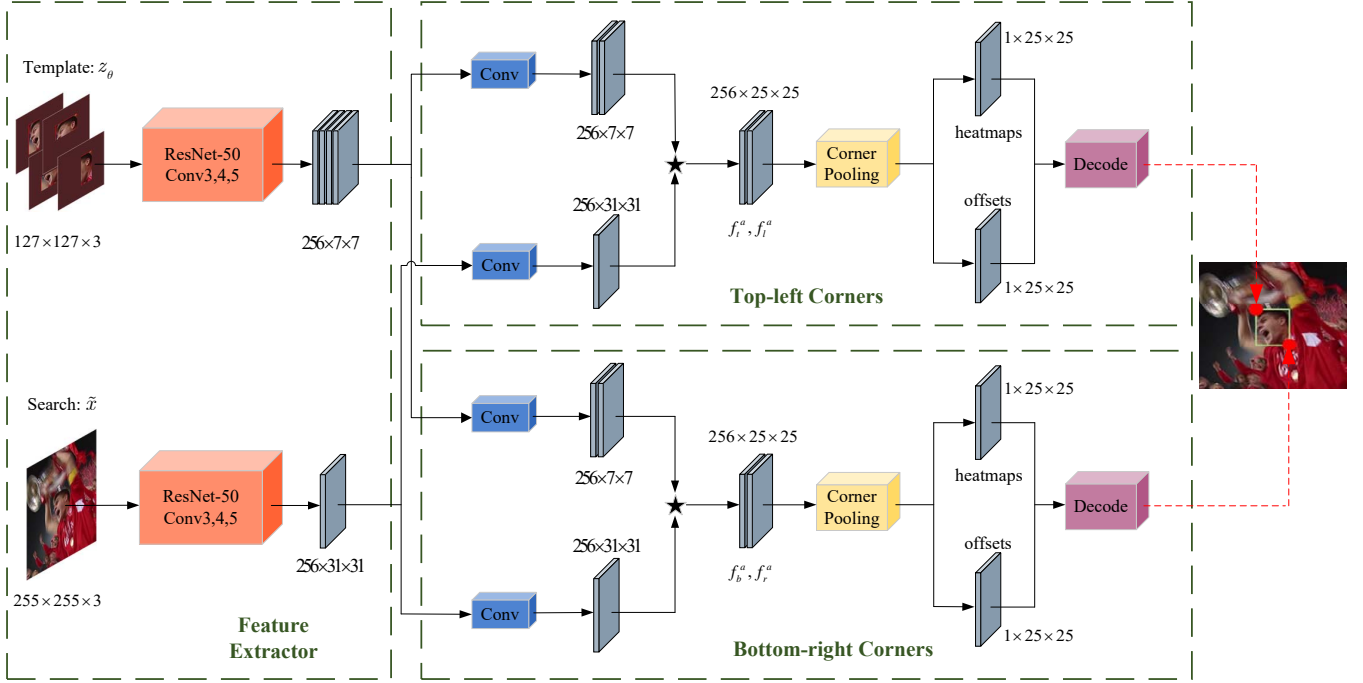


Fig. 2. The overview of our SiamCorners architecture, which includes the Siamese feature extractor followed by the top-left corner and bottom-right corner branches in parallel.  $\star$  denotes depth-wise correlation operation.  $z_\theta$  ( $\theta \in \{t, l, b, r\}$ ) denote the four boundary images cropped from the original image.

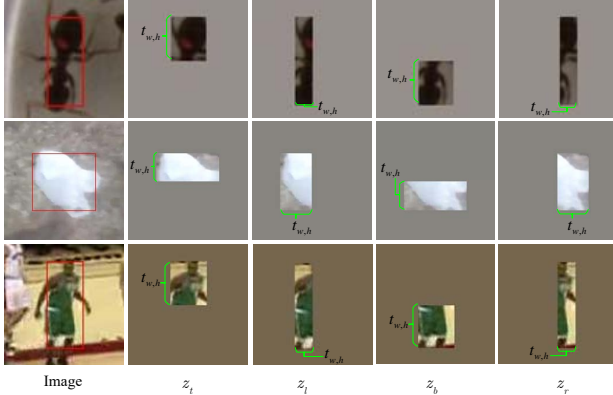


Fig. 3. Examples of four boundary images fed to the template branch. The boundary images are cropped from the  $t_{w,h}$  width or height of the red box of the input image.

We train the corner network following Law and Deng [20]. For each heatmap, there is only one ground-truth label for a corner as a positive location and the other locations are negative, which will make the network difficult to learn. For efficient training, we reduce the penalty when the negative locations are still within the radius of the positive location. We determine the radius by the at least  $d$  IoU (we set  $d = 0.5$  in all experiments) between a pair of corners and the ground-truth box. For each ground truth corner  $c(c_x, c_y) \in \mathbb{R}^2$  in the search image, we compute the location of corner heatmap  $\tilde{c}(\tilde{c}_x, \tilde{c}_y) = \lfloor \frac{c(c_x, c_y)}{s} \rfloor$ . Given the radius and location  $\tilde{c}$ , the amount of penalty reduction is calculated by 2D Gaussian,  $X_{xy} = \exp\left(-\frac{(x-\tilde{c}_x)^2 + (y-\tilde{c}_y)^2}{2\delta_c^2}\right)$ , where  $\delta_c$  is 1/3 of the radius. Again,  $X_{xy}$  denotes the ‘‘ground truth’’ heatmap designed

with unnormalized Gaussians. Let  $\hat{X}_{xy}$  be the prediction score of network at the location  $(x, y)$ . The training objective is a variant of focal loss [15]:

$$\mathcal{L}_{tr} = \frac{-1}{K} \sum_{xy} \begin{cases} (1 - \hat{X}_{xy})^\alpha \log(\hat{X}_{xy}) & \text{if } X_{xy} = 1 \\ (1 - X_{xy})^\beta (\hat{X}_{xy})^\alpha \log(1 - \hat{X}_{xy}) & \text{otherwise} \end{cases} \quad (1)$$

where  $\alpha$  and  $\beta$  denote the weighting factors which control the contribution of each corner (we set  $\alpha = 2$  and  $\beta = 4$  in all experiments).  $K$  is the number of candidate corners. In eq.(1), the  $(1 - X_{xy})$  term reduces the penalty when the predicted corners are within the radius of the positive location.

However, the output size of the network is usually smaller than the original image as the network becomes deeper. There will be a discretization error when mapping a location  $(m, n)$  of the original image to the location  $(\lfloor \frac{m}{s} \rfloor, \lfloor \frac{n}{s} \rfloor)$  in the heatmaps. Moreover, some precision will be lost in tracking when remapping locations from heatmaps to the original image. To address the discretization error caused by network stride, we introduce an offset network that is used to slightly adjust the corner offset before remapping the heatmap to the original image. The corner offset on the heatmap is calculated as:

$$\mathbf{o}_k = \left( \frac{m_k}{s} - \left\lfloor \frac{m_k}{s} \right\rfloor, \frac{n_k}{s} - \left\lfloor \frac{n_k}{s} \right\rfloor \right) \quad (2)$$

where  $\mathbf{o}_k$  is the corner offset,  $m_k$  and  $n_k$  are the  $k$ -th corner coordinates of the object in the original image. We employ the smooth L1 Loss [37] at corner locations:

$$\mathcal{L}_{off} = \frac{1}{K} \sum_{k=1}^K \text{SmoothL1Loss}(\mathbf{o}_k, \hat{\mathbf{o}}_k) \quad (3)$$

where  $\mathbf{o}_k$  and  $\hat{\mathbf{o}}_k$  are ‘‘ground truth’’ offset and offset network prediction, respectively. Therefore, the whole method optimizes the loss function  $\mathcal{L}$ , which is expressed as:

$$\mathcal{L} = \mathcal{L}_{tr} + \lambda \mathcal{L}_{off} \quad (4)$$

where  $\lambda$  (we set  $\lambda = 1$  in all experiments) is a hyper-parameter that controls the balance of the two loss terms.

### C. Corner Pooling

In this section, we describe how to use a modified corner pooling to predict corner heatmaps and offsets in Siamese architecture. The existing corner pooling is originally proposed for object detection, we make modifications to adapt it to object tracking. Given the template images  $z_\theta, \theta \in \{t, l, b, r\}$  and the search image  $\tilde{x}$ , the feature maps are efficiently denoted by parameterizing  $f^a(z_\theta) = \phi(z_\theta), a \in \{3, 4, 5\}$  and  $f^a(\tilde{x}) = \phi(\tilde{x})$ , where  $\phi$  is a convolutional neural network (CNN). Here,  $(f^3(z_\theta), f^3(\tilde{x}))$ ,  $(f^4(z_\theta), f^4(\tilde{x}))$ , and  $(f^5(z_\theta), f^5(\tilde{x}))$  denote the output features of *conv3*, *conv4*, and *conv5* branches in Fig. 2, respectively. Afterward, following the Siamese architecture, we aim to predict the depth-wise correlation feature:  $f_\theta^a = f^a(z_\theta) \star f^a(\tilde{x})$ . Therefore, the specific-target features are integrated into the SiamCorners network.

Here, we will describe in detail how to determine whether a pixel  $(i, j)$  is a bottom-right corner in correlation maps. On a  $\frac{H}{s} \times \frac{W}{s}$  correlation map, the corner pooling layer max-pools the all feature on  $f_b^a$  to get a feature vector  $b_{ij}^a$  from  $(i, 0)$  to  $(i, \frac{W}{s})$ , and max-pools the feature map  $f_r^a$  to get a feature vector  $r_{ij}^a$  from  $(0, j)$  to  $(\frac{H}{s}, j)$ . This calculation stage can be expressed as:

$$b_{ij}^a = \begin{cases} \max(f_{b_{ij}}^a, b_{i(j-1)}^a) & \text{if } 0 < j \leq \frac{W}{s} \\ f_{b_{i0}}^a & \text{otherwise} \end{cases} \quad (5)$$

$$r_{ij}^a = \begin{cases} \max(f_{r_{ij}}^a, r_{(i-1)j}^a) & \text{if } 0 < i \leq \frac{H}{s} \\ f_{r_{0j}}^a & \text{otherwise} \end{cases} \quad (6)$$

We then add  $b_{ij}^a$  and  $r_{ij}^a$  to get the final corner pooling result. Similarly, the pooling results  $t_{ij}^a$  and  $l_{ij}^a$  are obtained by max-pooling the feature vectors horizontally from right to left and vertically from bottom to top on the correlation feature maps  $f_{t_{ij}}^a$  and  $f_{l_{ij}}^a$ , respectively. Finally, the result of top-left corner pooling is obtained by the sum of the two pooling results.

The architecture of the whole corner pooling is shown in Fig.4. The starting part of the corner pooling module corresponding to the two inputs ( $f_b^a$  and  $f_r^a$ ) is a modified variant of the residual block [28], respectively. After the design of the residual block, the corner pooling feature is followed by a  $3 \times 3$  Conv-BN layer and its output is added to the two projection shortcuts. Finally, it predicts the corner heatmaps and offsets by feeding the modified residual feature through a  $3 \times 3$  Conv-BN-ReLU layer followed by a  $3 \times 3$  Conv-BN-ReLU-Conv layer, respectively. It is worth noting that the modified corner pooling and the original corner pooling in CornerNet [14] have different designs in the network structure and pooling mechanism. As shown in Fig. 4, the

modified corner pooling introduces bottom pooling and right pooling in the right and bottom boundary feature maps ( $f_r^a$  and  $f_b^a$ ) respectively. Both the bottom and right boundary features are followed by projection shortcuts to enhance the corner prediction accuracy. In contrast, the original corner pooling [14] for object detection task predicts the bottom-right corner by using the bottom-right corner pooling and a single projection shortcut in a unified feature map.

### D. Decoding for Corner Heatmaps and Offsets

In this section, we aim to predict the tracking bounding box by the given corner heatmaps and offsets. Following Section III-C, the final outputs of corner pooling have four parts: i) the top-left corner heatmaps. ii) the bottom-right corner heatmaps. iii) the top-left corner offsets. iv) the bottom-right corner offsets. After corner pooling, the top-left and bottom-right corner heatmaps are first followed by a sigmoid function, respectively. Then, a NMS operation is adopted to remove redundant corner locations. We further obtain the corresponding top-left corner location  $(X_{T_l}^a, Y_{T_l}^a) \in \mathbb{R}^{N \times 2}$  and bottom-right corner location  $(X_{B_r}^a, Y_{B_r}^a) \in \mathbb{R}^{N \times 2}$  by selecting the top  $N$  (we set  $N = 15$  in all experiments) corner scores in top-left and bottom-right corner heatmaps, respectively. To overcome the impact of network stride, we denote  $(\Delta X_{T_l}^a, \Delta Y_{T_l}^a)$  and  $(\Delta X_{B_r}^a, \Delta Y_{B_r}^a)$  as the corresponding offsets of the top-left corners and the bottom-right corners, respectively. Afterward, the refined top  $N$  corner sets  $COR^* = (\hat{X}_{T_l}^a, \hat{Y}_{T_l}^a, \hat{X}_{B_r}^a, \hat{Y}_{B_r}^a, S_{cor}^a)$  can be predicted by the following formula:

$$\begin{aligned} \hat{X}_{T_l}^a &= X_{T_l}^a + \Delta X_{T_l}^a, & \hat{Y}_{T_l}^a &= Y_{T_l}^a + \Delta Y_{T_l}^a \\ \hat{X}_{B_r}^a &= X_{B_r}^a + \Delta X_{B_r}^a, & \hat{Y}_{B_r}^a &= Y_{B_r}^a + \Delta Y_{B_r}^a \end{aligned} \quad (7)$$

$$S_{cor}^a = \frac{1}{2}(T_{l_{sco}}^a + B_{r_{sco}}^a)$$

where  $S_{cor}^a$  is a pair of corner scores which are the average score of the top-left corner score  $T_{l_{sco}}^a$  and bottom-right corner score  $B_{r_{sco}}^a$ . Here,  $COR^* \in \mathbb{R}^{N \times 5}$ . After the top  $N$  corners  $COR^*$  are obtained, we use multi-layer features fusion and some penalty strategies to select the optimal tracking corners.

### E. Multi-level Features Fusion and Corners Selection

**Features Fusion:** Intuitively, visual tracking requires rich feature representations that include features from low-level to high-level, and resolutions from fine to coarse and scales from small to large [1]. To this end, we introduce a multi-layer features fusion strategy to help SiamCorners benefit from low-level features and high-level features. Following Section III-D, we concatenate  $\hat{X}_{T_l}^a, \hat{Y}_{T_l}^a, \hat{X}_{B_r}^a, \hat{Y}_{B_r}^a$  and  $S_{cor}^a$  and use a simple mapping to get the corner locations  $X_{T_l}, Y_{T_l}, X_{B_r}, Y_{B_r}$  and  $S_{cor}$  corresponding to the original image, respectively. Therefore, the corner sets in the original image can be denoted as a  $3N \times 5$  dimension term  $COR = (X_{T_l}, Y_{T_l}, X_{B_r}, Y_{B_r}, S_{cor})$ .

**Corners Selection:** After obtaining the corners  $COR$ , we introduce a post-process strategy to make the SiamCorners more suitable for tracking tasks. In visual tracking, the target in nearby frames almost does not have large variations in size

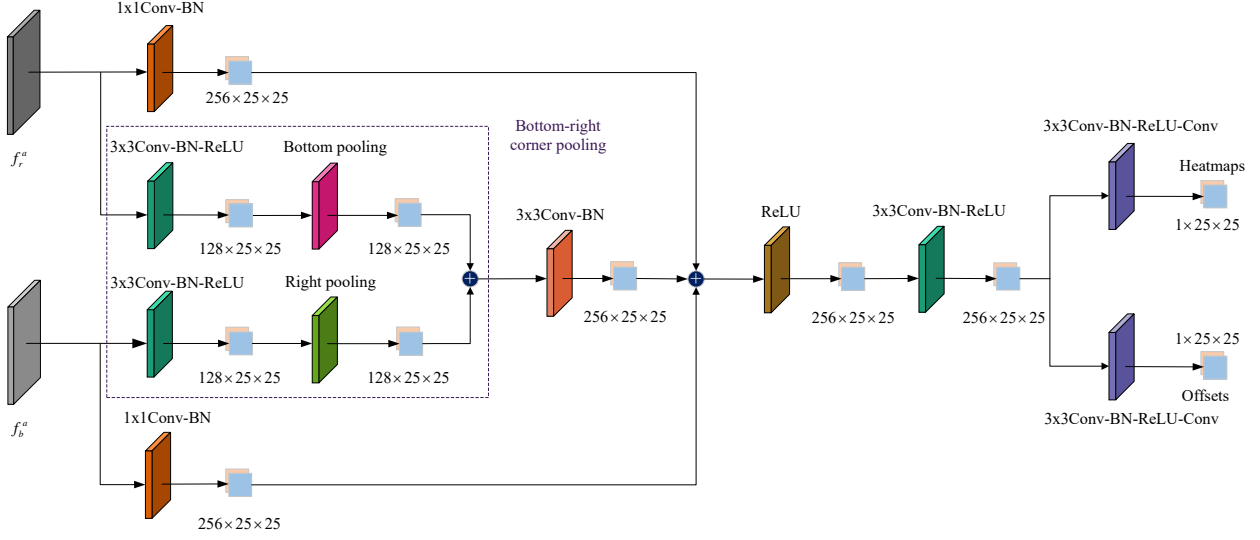


Fig. 4. The corner pooling module starts with the two feature maps, which are modified from the residual block. The bottom pooling and right pooling use the max-pools methods of eq.5 and 6, respectively. The entire corner pooling layer has two branches that predict the target’s heatmaps and offsets, respectively. In the figure, the  $3 \times 3$  Conv-BN-ReLU module consists of a  $3 \times 3$  convolution layer, a Batch Normalization layer, and a ReLU layer. Similarly, the  $3 \times 3$  Conv-BN-ReLU-Conv module consists of the Conv-BN-ReLU module and a  $1 \times 1$  convolution layer.

and shape. We calculate the overall scales  $c_{sc}$  of the corners by the following formula:

$$c_{sc}^2 = (w + p)(h + p) \quad (8)$$

where  $w = X_{B_r} - X_{T_l}$  and  $h = Y_{B_r} - Y_{T_l}$  denote the width and height of the target, and  $p = \frac{1}{2}(w + h)$  denotes the padding. Similarly, the same principle is used to calculate the overall scales  $\hat{c}_{sc}^2$  of corners at the last frame. We use a penalty term to suppress large changes in the size and ratio of the target:

$$C_{penalty} = e^{\eta * \max(\frac{c_r}{\hat{c}_r}, \frac{\hat{c}_r}{c_r}) * \max(\frac{c_{sc}}{\hat{c}_{sc}}, \frac{\hat{c}_{sc}}{c_{sc}})} \quad (9)$$

where  $c_r$  and  $\hat{c}_r$  denote the corners’ ratio of height and width in the current frame and the last frame, respectively. Here,  $\eta$  is a hyper-parameter. Therefore, the score  $S_{pen}$  of each corner in the  $COR$  is calculated by the weighted penalty as:  $S_{pen}^u = C_{penalty}^u * S_{cor}^u$ ,  $u = 1, 2, \dots, 3N$ . Unlike SiamRPN [1], [11], we further introduce a novel index term  $U(u)$ ,  $u = 1, 2, \dots, 3N$ , which represents the index of the variation degree of each corner in  $COR$  from large to small compared with the last frame. The metric of variation degree is determined by the sum of the absolute values of the difference between the center, width, and height of the target between two adjacent frames. For example, when  $u = 1$ ,  $S_{pen}^{U(1)}$  means that the  $U(1)$ -th corner in the corner sets  $COR$  has the largest motion change relative to the last frame. Finally, the scores  $S_{final}^{U(u)}$  of each corner can be expressed as:

$$S_{final}^{U(u)} = S_{pen}^{U(u)} * (1 - \gamma) + S_{han}^u * \gamma \quad (10)$$

where  $\gamma$  is a hyper-parameter that controls the balance between the two score terms.  $S_{han}^u$  is a variant of the Hanning window since we only take the monotonically increasing part of the Hanning window. After obtaining the final score  $S_{final}$  of

---

#### Algorithm 1 Tracking as one-shot

---

**Input:** initial template images:  $z_\theta$ ; search image:  $\tilde{x}$ ;

**Output:** Tracking bounding box  $x^*$

---

- 1:  $f^a(z_\theta) \leftarrow \phi(z_\theta)$ ; # Feed the template images into CNN.
  - 2: **repeat**
  - 3:  $f^a(\tilde{x}) \leftarrow \phi(\tilde{x})$ ;
  - 4:  $f_t^a, f_l^a, f_b^a, f_r^a \leftarrow f^a(z_\theta) \star f^a(\tilde{x})$  # Depth-wise correlation
  - 5:  $t^a, l^a, b^a, r^a \leftarrow CornerPooling(f_t^a, f_l^a, f_b^a, f_r^a)$  # Section III-C
  - 6:  $COR^* \leftarrow Decoding(t^a, l^a, b^a, r^a)$  # Decoding process (Section III-D)
  - 7: **for**  $a = 3, 4, 5$  **do**
  - 8:  $COR \leftarrow (\hat{X}_{T_l^a}, \hat{Y}_{T_l^a}, \hat{X}_{B_r^a}, \hat{Y}_{B_r^a}, S_{cor}^a)$  # Concatenation
  - 9: **end for**
  - 10:  $COR_{final} \leftarrow COR$  # Update corner scores using eq.(8)-(10)
  - 11:  $x^* \leftarrow PostProcessing(COR_{final})$  # Selecting a pair of corners and using a linear interpolation operation
  - 12: **until** end of sequence
- 

each corner, the expression of the corner set  $COR$  is replaced by  $COR_{final} = (X_{T_l}, Y_{T_l}, X_{B_r}, Y_{B_r}, S_{final})$ . Afterwards, we pick the corners with the highest score in the  $COR_{final}$  as the final tracking bounding box. Furthermore, a linear interpolation operation is used to update the target size so that the shape can be changed smoothly.

Finally, we summarize the SiamCorners in algorithm 1.

#### IV. EXPERIMENTAL RESULTS

Our method is implemented in Python using PyTorch. On a single NVIDIA RTX 2080 GPU, SiamCorners achieves a tracking speed of 42 FPS by employing ResNet-50 as backbone. To facilitate further research, the

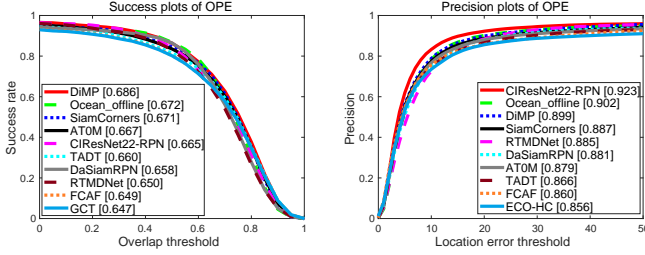


Fig. 5. Experimental results of the methods on the OTB100 dataset.

completed training and testing code will be released at <https://github.com/yangkai12/SiamCorners>.

### A. Training Details

We use ResNet-50 pre-trained on ImageNet [38] as the backbone network. The training splits of the GOT-10k [39], LaSOT [6], COCO [34], YouTube-BoundingBoxes [40], ImageNet VID [38], and ImageNet DET datasets are used to train our network to learn how to measure the similarity of input image pairs in visual tracking. In both training and testing phases, we set the sizes of the template image and test image to 127 pixels and 255 pixels, respectively. To reduce overfitting, we implemented some data augmentation strategies in the input image pairs, such as random horizontal flipping, random color jittering, random scaling, random cropping, adjusting the brightness and contrast of an image. We use stochastic gradient descent (SGD) with a batchsize of 58 to optimize the overall training loss from eq. (4). We set a learning rate of 0.001 for earlier 5 epochs to train the top-left branch and bottom-right branch. For the latter 15 epochs, the whole network is trained in an end-to-end manner, where the learning rate decays exponentially from 0.005 to 0.0005, while weight decay and momentum are set to 0.0001 and 0.9, respectively. Unlike SiamRPN++ [1], we do not have a finetune backbone operation in the last 10 epochs, which will increase training time exponentially.

### B. Testing Metrics

In this experiment, we evaluate our method on tracking benchmarks OTB100 [5], UAV123 [7], LaSOT [6], NFS30 [8], and TrackingNet [9]. We use success scores and precision scores to evaluate the performance of the trackers on OTB100, UAV123 and NFS30. The success scores are defined as the percentage of success frames when the overlap threshold between the predicted result and the ground truth varies from 0 to 1, where a success frame means that the tracking result is larger than a given threshold. For a fair comparison, all trackers are ranked by the area under curve (AUC) of the success plot. The precision score is computed as the percentage of frames, which is defined that the Euclidean distance between the center of the prediction target and the ground truth is within a given distance threshold. Here, the different distance thresholds are used to evaluate all trackers. For the LaSOT and TrackingNet datasets, we use the success score, precision score, and normalized precision score to evaluate the performance of all trackers,

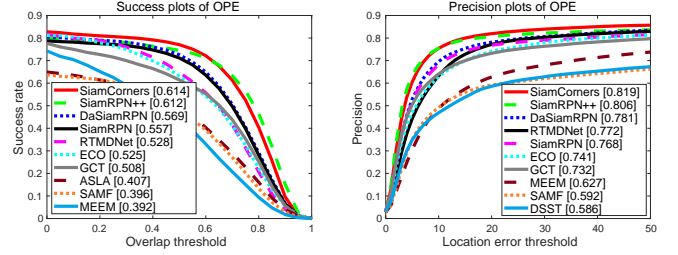


Fig. 6. Experimental results of the methods on the UAV123 dataset.

where the definitions of success and precision score are the same as previously described. Meanwhile, the normalized precision is defined as the normalization of precision over the size of the ground truth bounding box. The trackers are then ranked by AUC score with normalized precision between 0 and 0.5. Generally, the AUC score of success plot is mainly used to rank all trackers.

### C. State-of-the-art Comparison

**OTB100 [5]:** OTB100 is one of the most widely used benchmark datasets in tracking, which includes 100 fully labeled video sequences. The Area Under Curve (AUC) of success plot is employed to compare SiamCorners and nine state-of-the-art trackers including TADT [41], GCT [42], CIResNet-22-RPN [3], RT-MDNet [43], SiamRPN [11], DaSiamRPN [10], FCAF [29], Ocean [18], ATOM [44] and DIMP [45]. As shown in Fig. 5, our method performs better than the top tracker, such as ATOM, and slightly lower than its optimized version, DIMP. Our method outperforms the anchor-based trackers CIResNet22-RPN and DaSiamRPN with relative gains of 0.6% and 1.3% in terms of success score, respectively. Further, our method achieves a gain of 2.2% in AUC over FCAF, while on par with offline anchor-free tracker Ocean. This shows that SiamCorners can achieve the state-of-the-art results by eliminating the anchor boxes.

**UAV123 [7]:** We evaluate the proposed SiamCorners method on the UAV123 dataset in Fig. 6. The dataset is collected from a low-altitude UAV, which contains 123 video sequences and more than 110K frames in total. In this dataset, we compare our method with recent trackers, *i.e.* SiamRPN++ [1], DaSiamRPN [10], SiamRPN [11], RTMDNet [43], ECO [46], GCT [42], ASLA [58], SAMF [59], MEEM [60], and DSST [61]. In terms of the success score and precision score, SiamRPN++ achieves scores of 61.2% and 80.6%, respectively. Our approach outperforms the top tracker SiamRPN++ with relative gains of 0.2% and 1.3% in terms of the success score and precision score, respectively. It shows that our method achieves superior accuracy when compared with the recent methods based on anchor boxes.

In Fig. 7, we analyze the performance of the trackers under 12 challenging factors in tracking scenarios, such as lighting changes, scale variation, and aspect ratio change. These attributes are used to evaluate the performance of the trackers against different aspects. For clarity, the legend only shows the top 10 AUC scores in each plot. We observe that the proposed

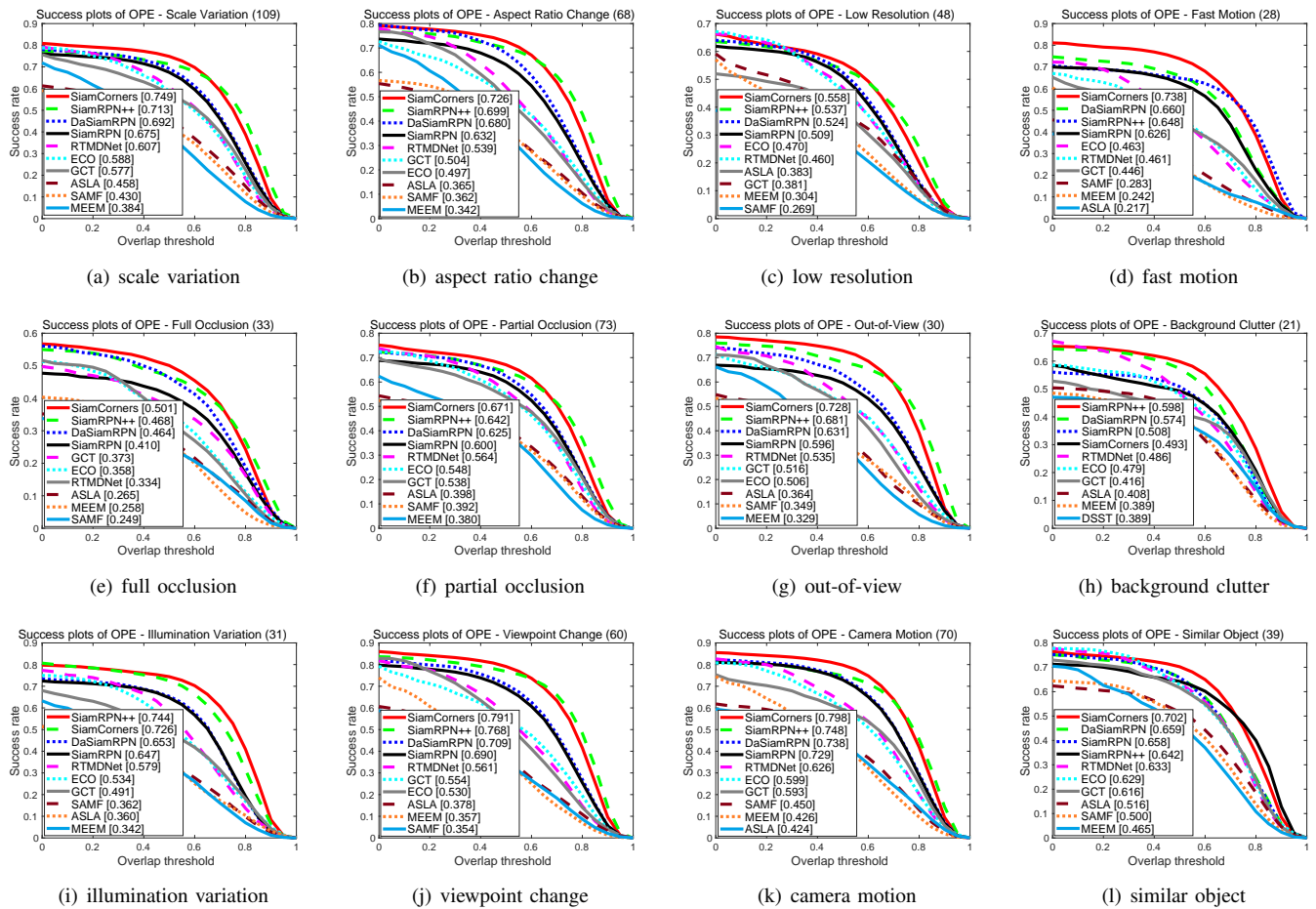


Fig. 7. The success plots on UAV123 dataset for 12 challenging factors: scale variation, aspect ratio change, low resolution, fast motion, full occlusion, partial occlusion, out-of-view, background clutter, illumination variation, viewpoint change, camera motion, and similar object. The legend shows the AUC score for each method.

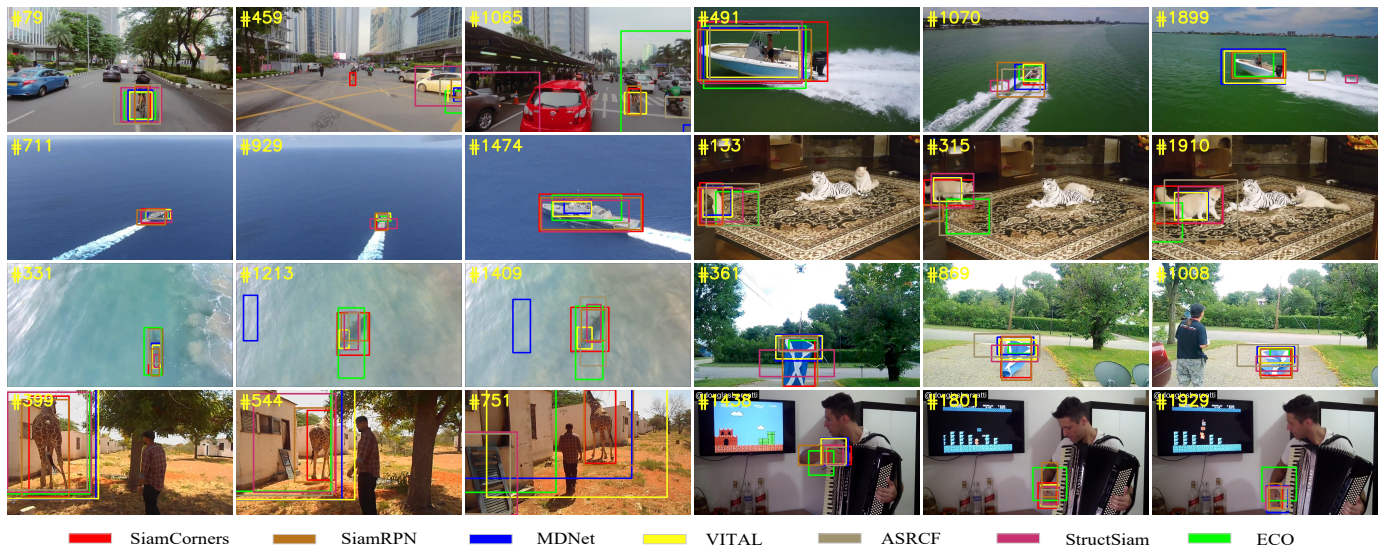


Fig. 8. Visualization results of different methods on LaSOT dataset (from left to right and top to bottom: *bicycle-9*, *boat-4*, *boat-17*, *cat-3*, *crocodile-10*, *flag-5*, *giraffe-10* and *hand-16*).

SiamCorners tracker handles well in scale variation, aspect ratio change, low resolution, fast motion, full occlusion, partial

occlusion, out-of-view, viewpoint change, camera motion, and similar object. In particular, the proposed SiamCorners tracker



TABLE I

A COMPARISON BETWEEN THE SIAMCORNERS AND STATE-OF-THE-ART TRACKERS ON TRACKINGNET DATASET IN TERMS OF AUC SCORE, PRECISION SCORE (P) AND NORMALIZED PRECISION SCORE ( $P_{norm}$ )

|                | ECO [46] | SiamFC [25] | CFNet [26] | MDNet [47] | GFS-DCF [48] | UPDT [49] | DaSiamRPN [10] | UpdateNet [50] | ATOM [44] | SiamCorners |
|----------------|----------|-------------|------------|------------|--------------|-----------|----------------|----------------|-----------|-------------|
| AUC(%)         | 55.4     | 57.1        | 57.8       | 60.6       | 60.9         | 61.1      | 63.8           | 67.7           | 70.3      | 69.5        |
| P(%)           | 49.2     | 53.3        | 53.3       | 56.5       | 56.6         | 55.7      | 59.1           | 62.5           | 64.8      | 64.7        |
| $P_{norm}$ (%) | 61.8     | 66.3        | 65.4       | 70.5       | 71.8         | 70.2      | 73.3           | 75.2           | 77.1      | 76.3        |

TABLE II

A COMPARISON BETWEEN THE SIAMCORNERS AND STATE-OF-THE-ART TRACKERS ON NFS DATASET.

|        | BACF [51] | SRDCF [52] | FCNT [53] | DaSiam-RPN [54] | HDT [55] | MDNet [47] | MKCF [56] | ECO [46] | CCOT [57] | SiamRPN++ [1] | SiamCorners |
|--------|-----------|------------|-----------|-----------------|----------|------------|-----------|----------|-----------|---------------|-------------|
| AUC(%) | 34.2      | 35.3       | 39.3      | 39.5            | 40.0     | 42.5       | 45.5      | 47.0     | 49.2      | 49.2          | 53.7        |
| FPS    | 35.3      | 13.0       | 3.0       | 160.0           | -        | 1.0        | 30.0      | 8.0      | 0.3       | 35.0          | 42.0        |

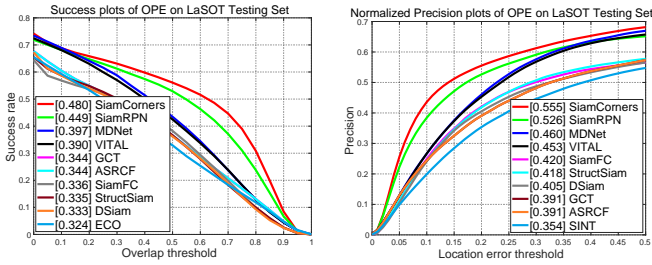


Fig. 9. Experimental results of the methods on the LaSOT dataset.

has a 7.8% higher AUC score than DaSiamRPN when tracking targets suffer from fast motion. These results show that the introduced SiamCorners method has the potential to solve the fast motion problem.

**TrackingNet [9]:** TrackingNet is a recently proposed large-scale dataset in the wild, which provides more than 30K videos and 14 million bounding box annotations. We evaluate SiamCorners on its test split with 511 videos. Table I shows the results of several state-of-the-art trackers, which are obtained by an online evaluation server. Among the compared approaches, ATOM achieves the best tracking result with an AUC score of 70.3%. Our SiamCorners achieves an AUC score of 69.5%, competitive with the other state-of-the-art trackers.

**LaSOT [6]:** To further validate the effectiveness of the SiamCorners tracker, we evaluate our tracker on LaSOT [6] dataset. LaSOT is a recently proposed tracking benchmark dataset, which contains 1400 sequences with an average video length of 2500 frames. We evaluate SiamCorners on its test split containing 280 videos. Furthermore, we compare our SiamCorners method with some representative trackers, that is, SiamRPN [11], MDNet [47], VITAL [62], GCT [42], ASRCF [63], SiamFC [25], StructSiam [64], DSiam [65], SINT [66], and ECO [46]. As shown in Fig. 9, SiamCorners achieves the best performance in both success plot and normalized precision plot. Our approach outperforms MDNet with relative gains of 8.0% and 9.3% in terms of success score and normalized precision score, respectively.

Figure 8 shows the visualization results of the trackers

on the LaSOT dataset. Although these methods generally track targets accurately, they do not perform well under some challenging factors. The ECO and ASRCF based on correlation filters do not perform well in testing sequences with aspect ratio change (bicycle-9, cat-3, and giraffe-10). The MDNet and VITAL trackers are all built on classification component of CNN, which are prone to drift off the target when tracking targets undergo illumination variation (bicycle-9), partial occlusion (giraffe-10), and aspect ratio change (crocodile-10). The SiamRPN and StructSiam trackers that follow the common Siamese architecture are less effective in addressing sequences with deformation (hand-16) and scale variation (boat-4, cat-3, and giraffe-10) attributes. In particular, the tracker SiamRPN based on anchor boxes cannot handle the scale variation problem well since it is difficult to handle the scale variation of the target by setting fixed ratio anchor boxes. Overall, the tracking results of the proposed SiamCorners performs well on most sequences. This qualitatively demonstrates the superiority of SiamCorners compared with other state-of-the-art methods.

**NFS30 [8]:** The NFS30 dataset contains 100 videos and a length of 380K frames in total, which are mainly captured by high frame rate cameras in real-world scenarios. We evaluate the SiamCorners tracker on the 30 FPS version of this dataset, which contains some challenging factors for fast-moving targets. From Table II, we can observe that SiamCorners achieves 4.5% higher than SiamRPN++ in terms of the AUC score. Furthermore, our method achieves a speed gain of 7 FPS compared with SiamRPN++, which verifies that the speed gain can be obtained by discarding anchors to reduce network parameters.

#### D. Ablation Study

To verify the effectiveness of each component in the proposed approach, we perform an ablation study on the UAV123 and LaSOT datasets in terms of AUC score of success plot. To be fair, we use the same training datasets and parameter settings in the training step for different components.

**Backbone Architecture.** Different convolutional layers of the backbone network produce different feature representations

TABLE III  
ABLATION STUDY OF THE SIAMCORNERS TRACKER ON UAV123 AND LASOT.

| Backbone  | L3 | L4 | L5 | Offset | MCP | OCP | PPF | OPF | FRT | OUT | UAV123 | LaSOT |
|-----------|----|----|----|--------|-----|-----|-----|-----|-----|-----|--------|-------|
| ResNet-50 | ✓  |    |    | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.527  | 0.380 |
|           |    | ✓  |    | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.587  | 0.432 |
|           |    |    | ✓  | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.523  | 0.373 |
|           | ✓  | ✓  |    | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.556  | 0.401 |
|           | ✓  |    | ✓  | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.539  | 0.391 |
|           |    | ✓  | ✓  | ✓      | ✓   |     | ✓   |     | ✓   |     | 0.601  | 0.447 |
|           | ✓  | ✓  | ✓  | ✓      |     |     | ✓   |     | ✓   |     | 0.593  | 0.455 |
|           | ✓  | ✓  | ✓  | ✓      | ✓   | ✓   |     |     | ✓   | ✓   | 0.600  | 0.463 |
|           | ✓  | ✓  | ✓  | ✓      | ✓   |     | ✓   |     |     | ✓   | 0.551  | 0.411 |
|           | ✓  | ✓  | ✓  |        | ✓   | ✓   |     | ✓   |     | ✓   | 0.599  | 0.441 |
|           | ✓  | ✓  | ✓  | ✓      | ✓   | ✓   |     | ✓   |     | ✓   | 0.614  | 0.480 |

Note: L3, L4, and L5 represent the *conv3*, *conv4*, and *conv5* outputs of ResNet-50, respectively. Offset represents whether to add an offset network to the convolutional network to predict the corner offsets.

and parameters, which will affect the speed, memory usage, and performance of the tracker. As shown in Table III, we empirically observe that *conv4* branch achieves the best AUC score in a single-level feature, while shallower or deeper layers obtain performance drops. Furthermore, the performance will be further improved by combining *conv4* branch and *conv5* branch, while the performance is not improved when combining the other two branches. Finally, SiamCorners achieves the best result that produces a 0.614 AUC score on UAV123 through aggregating all three layers, which is 2.7% higher than the single-layer baseline.

**Offset.** We investigate the impact of the offset network by excluding it from our SiamCorners framework. As shown in Table III, the proposed offset operation gains 1.5% improvement on UAV123 and 3.9% improvement on LaSOT, which demonstrates the importance of offset network. This happens because the introduced offset network can alleviate the impact of the network stride and thus improve the performance.

**Corner Pooling.** We investigate the impact of the original corner pooling (OCP) and the modified corner pooling (MCP) on the final tracking results. Note that the original corner pooling operation only performs corner prediction on a single feature map. We feed the feature maps of the top and left boundary images to the top-left corner pooling of OCP and the feature maps of the bottom and right boundary images to the bottom-right corner pooling of OCP in the ablation experiment. Table III shows that introducing a MCP gains 2.1% improvement on UAV123 and 2.5% improvement on LaSOT. This shows that the MCP operation is a key component in the SiamCorners framework. We further investigate the proposed penalty function (PPF) and the original penalty function (OPF) on the tracking performance. Using a PPF to select the final tracking box yields 1.4% improvement on UAV123 and 1.7% improvement on LaSOT. Finally, we analyze the impact of the four respective templates (FRT) and one unified template (OUT) on the tracking performance. We use the OCP operation

TABLE IV  
COMPONENT-WISE COMPUTATION TIME OF THE SIAMCORNERS TRACKER ON UAV123 DATASET

| Component | C3   | C4   | C5   | Off  | Cor  | Dec  | Pos  |
|-----------|------|------|------|------|------|------|------|
| Time (ms) | 2.30 | 4.43 | 5.69 | 0.63 | 4.38 | 3.29 | 0.82 |

Note: C3, C4, and C5 represent calculation time that the testing images are fed to *conv3*, *conv4*, and *conv5* of ResNet-50, respectively. Off and Cor represent the offset predictions of the corners and corner pooling operations, respectively. Dec and Pos represent decoding operations and post-processing operations that correspond to Section III-D and Section III-E, respectively.

since OUT only generates a single feature map as input for corner pooling. Further, FRT obtains a 6.3% improvement on UAV123 and a 6.9% improvement on LaSOT. This suggests the FRT adds some performance improvements to our tracker.

**Component-wise Time.** We investigate the component-wise computation time of the SiamCorners tracker on the UAV123 dataset, shown in Table IV. The last residual block *conv5* of ResNet-50 takes the most computational time. Since the number of parameters on the feature extractor will be increased by using a deeper network, which increases the running time. It is worth noting that the calculation time of corner pooling ranks third. In the following research, we can design a more efficient pooling operation to reduce the running time.

## V. CONCLUSIONS

In this paper, we propose a novel tracking architecture, SiamCorners, which does not require multi-scale testing or pre-defined anchor boxes with different aspect ratios. SiamCorners directly tracks the targets as a pair of corners. Our architecture integrates layer-wise features in deep networks. By exploring a new penalty function, we can constrain the variation of center, width, and height of the bounding boxes in adjacent frames. We evaluate SiamCorners on OTB100, UAV123, NFS30, LaSOT, and TrackingNet datasets, which demonstrates its effectiveness and efficiency. We hope that our

work will lead to further exploration for anchor-free methods in visual tracking.

#### ACKNOWLEDGMENT

This research was supported in part by Special Research project on COVID-19 Prevention and Control of Guangdong Province (Grant No. 2020KZDZDX1227), in part by the Shenzhen Research Council (Grant No. JCYJ20170413104556946 and No. JCYJ20170413105929681), in part by the Natural Science Foundation of China under Grant No. 62006060, Grant No. U2013210, Grant No. 62002241 and Grant No. 61972112, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2021B1515020088. This work was also supported in part by funding (Grant No. 2019-INT021) from Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS).

#### REFERENCES

- [1] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.
- [2] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1328–1338.
- [3] Z. Zhang and H. Peng, "Deeper and wider siamese networks for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4591–4600.
- [4] Z. Chen, B. Zhong, G. Li, S. Zhang, and 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.
- [5] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.
- [6] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5374–5383.
- [7] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 445–461.
- [8] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1125–1134.
- [9] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 300–317.
- [10] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 101–117.
- [11] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.
- [12] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9627–9636.
- [13] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9759–9768.
- [14] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 734–750.
- [15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2980–2988.
- [16] K. Kim and H. S. Lee, "Probabilistic anchor assignment with iou prediction for object detection," in *European Conference on Computer Vision*, 2020.
- [17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
- [18] Z. Zhang, H. Peng, J. Fu, B. Li, and W. Hu, "Ocean: Object-aware anchor-free tracking," *European Conference on Computer Vision*, August 2020.
- [19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.
- [20] H. Law, Y. Teng, O. Russakovsky, and J. Deng, "Cornernet-lite: Efficient keypoint based object detection," *arXiv preprint arXiv:1904.08900*, 2019.
- [21] W. Ruan, J. Chen, Y. Wu, J. Wang, C. Liang, R. Hu, and J. Jiang, "Multi-correlation filters with triangle-structure constraints for object tracking," *IEEE Transactions on Multimedia*, vol. 21, no. 5, pp. 1122–1134, 2018.
- [22] C. Liu, P. Liu, W. Zhao, and X. Tang, "Robust tracking and redetection: Collaboratively modeling the target and its context," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 889–902, 2017.
- [23] Y. Qi, S. Zhang, W. Zhang, L. Su, and M.-H. Yang, "Learning attribute-specific representations for visual tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8835–8842.
- [24] Q. Liu, X. Li, Z. He, N. Fan, D. Yuan, W. Liu, and Y. Liang, "Multi-task driven feature models for thermal infrared tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 604–11 611.
- [25] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 850–865.
- [26] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and 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.
- [27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
- [28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [29] G. Han, H. Du, J. Liu, N. Sun, and X. Li, "Fully conventional anchor-free siamese networks for object tracking," *IEEE Access*, vol. 7, pp. 123 934–123 943, 2019.
- [30] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 483–499.
- [31] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6569–6578, 2019.
- [32] A. Rashwan, R. R. Agarwal, A. Kalra, and P. Poupart, "Matrixnets: A new scale and aspect ratio aware architecture for object detection," *arXiv: Computer Vision and Pattern Recognition*, 2020.
- [33] Z. Dong, G. Li, F. W. Yue Liao, P. Ren, and C. Qian, "Centripetalnet: Pursuing high-quality keypoint pairs for object detection," pp. 10 519–10 528, 2020.
- [34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.
- [35] Z. He, S. Yi, Y. Cheung, X. You, and Y. Y. Tang, "Robust object tracking via key patch sparse representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 354–364, 2017.
- [36] Z. He, X. Li, X. You, D. Tao, and Y. Y. Tang, "Connected component model for multi-object tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3698–3711, 2016.
- [37] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.
- [38] 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," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [39] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

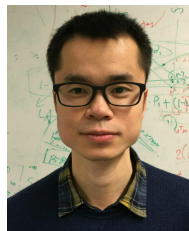
- [40] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5296–5305.
- [41] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, "Target-aware deep tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1369–1378.
- [42] J. Gao, T. Zhang, and C. Xu, "Graph convolutional tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4649–4659.
- [43] I. Jung, J. Son, M. Baek, and B. Han, "Real-time mdnet," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 83–98.
- [44] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4660–4669.
- [45] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte, "Learning discriminative model prediction for tracking," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6182–6191, 2019.
- [46] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6638–6646.
- [47] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.
- [48] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler, "Joint group feature selection and discriminative filter learning for robust visual object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7950–7960.
- [49] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 483–498.
- [50] L. Zhang, A. Gonzalez-Garcia, J. v. d. Weijer, M. Danelljan, and F. S. Khan, "Learning the model update for siamese trackers," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4010–4019.
- [51] H. Kiani Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1135–1143.
- [52] A. Lukežič, T. Vojř, L. Čehovin, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability [j]," *International Journal of Computer Vision*, vol. 126, no. 7, pp. 671–688, 2018.
- [53] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3119–3127.
- [54] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *European Conference on Computer Vision*, 2018.
- [55] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged deep tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4303–4311.
- [56] M. Tang, B. Yu, F. Zhang, and J. Wang, "High-speed tracking with multi-kernel correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4874–4883.
- [57] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 472–488.
- [58] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1822–1829.
- [59] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.
- [60] J. Zhang, S. Ma, and S. Sclaroff, "Meem: robust tracking via multiple experts using entropy minimization," in *European Conference on Computer Vision*. Springer, 2014, pp. 188–203.
- [61] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.
- [62] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8990–8999.
- [63] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li, "Visual tracking via adaptive spatially-regularized correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4670–4679.
- [64] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, "Structured siamese network for real-time visual tracking," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 351–366.
- [65] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1763–1771.
- [66] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1420–1429.



**Kai Yang** is now pursuing the Ph.D. degree with the Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His interests include deep learning, machine learning and visual tracking.



**Zhenyu He** received his Ph.D. degree from the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 2007. From 2007 to 2009, he worked as a postdoctoral researcher in the department of Computer Science and Engineering, Hong Kong University of Science and Technology. He is currently a full professor in the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China. His research interests include machine learning, computer vision, image processing and pattern recognition.



**Wenjie Pei** is currently an Assistant Professor with the Harbin Institute of Technology, Shenzhen, China. He received the Ph.D. degree from the Delft University of Technology, the Netherlands in 2018, working with Dr. Laurens van der Maaten and Dr. David Tax. Before joining Harbin Institute of Technology, he was a Senior Researcher on Computer Vision at Tencent Youtu X-Lab. In 2016, he was a visiting scholar with the Carnegie Mellon University. His research interests lie in Computer Vision and Pattern Recognition including sequence modeling, deep learning, video/image captioning, etc.



**Zikun Zhou** received his Master degree from Harbin Institute of Technology in 2018. He is currently a Ph.D. candidate in school of Computer Science and Technology at Harbin Institute of Technology, Shenzhen. His research interests include computer vision and machine learning.



**Xin Li** is a postdoc researcher with the AI center at the Peng Cheng Laboratory. He received the Ph.D. degree in computer applied technology from Harbin Institute of Technology, Shenzhen, China in 2020. His research interests include visual tracking, machine learning, and computer vision.



**Di Yuan** received the M.S. degree in Applied Mathematics from Harbin Institute of Technology, Shenzhen, China, in 2017, where he is currently pursuing the Ph.D. degree in Computer Science and Technology. His current research interests include object tracking, image processing, and self-supervised learning.



**Haijun Zhang (M'13)** received the B.Eng. and Master's degrees from Northeastern University, Shenyang, China, and the Ph.D. degree from the Department of electronic Engineering, City University of Hong Kong, Hong Kong, in 2004, 2007, and 2010, respectively. He was a Post-Doctoral Research Fellow with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, from 2010 to 2011. Since 2012, he has been with the Shenzhen Graduate School, Harbin Institute of Technology, China, where he is currently a Professor of Computer Science. His current research interests include multimedia data mining, machine learning, computational advertising, and service computing. Prof. Zhang is currently an Associate Editor of *Neurocomputing*, *Neural Computing and Applications*, and *Pattern Analysis and Applications*.