SiamMan: Siamese Motion-aware Network for Visual Tracking

Wenzhang Zhou[1][*], Longyin Wen[2*], Libo Zhang[3][†]
Dawei Du[4], Tiejian Luo[1], Yanjun Wu[3]
[1]University of Chinese Academy of Sciences, China,
[2]JD Finance America Corporation, Mountain View, USA,
[3]Institute of Software Chinese Academy of Sciences, China,
[4]University at Albany, State University of New York, USA.

zhouwenzhang19@mails.ucas.ac.cn, longyin.wen.cv@gmail.com
{libo,yanjun}@iscas.ac.cn, cvdaviddo@gmail.com

Abstract

In this paper, we present a novel siamese motion-aware network (SiamMan) for visual tracking, which consists of the siamese feature extraction subnetwork, followed by the classification, regression, and localization branches in parallel. The classification branch is used to distinguish the foreground from background, and the regression branch is adopt to regress the bounding box of target. To reduce the impact of manually designed anchor boxes to adapt to different target motion patterns, we design the localization branch, which aims to coarsely localize the target to help the regression branch to generate accurate results. Meanwhile, we introduce the global context module into the localization branch to capture long-range dependency for more robustness in large displacement of target. In addition, we design a multi-scale learnable attention module to guide these three branches to exploit discriminative features for better performance. The whole network is trained offline in an end-to-end fashion with large-scale image pairs using the standard SGD algorithm with back-propagation. Extensive experiments on five challenging benchmarks, i.e., VOT2016, VOT2018, OTB100, UAV123 and LTB35, demonstrate that SiamMan achieves leading accuracy with high efficiency. Code can be found at `https://isrc.iscas.ac.cn/gitlab/research/siamman.`

1. Introduction

Visual object tracking is a hot research direction with a wide range of applications, such as surveillance, autonomous driving, and human-computer interaction. Although significant progress has been made in recent years, it is still a challenging task due to various factors, including occlusion, abrupt motion, and illumination variation.

Modern visual tracking algorithms can be roughly divided into two categories: (1) the correlation filter based approaches [9, 26, 34, 40], and (2) the deep convolution network based approaches [35, 24, 38, 10]. The correlation filter (CF) based approach trains a regressor for tracking using circular correlation via Fast Fourier Transform (FFT). With the arrival of the deep convolution network, some researchers use offline learned deep features [11, 9, 18] to improve the accuracy. Considering efficiency, those trackers abandon model update in tracking process, which greatly harms the accuracy and generally performs worse than the CF based approaches.

Recently, the deep Siamese-RPN method [25] is presented, which formulates the tracking task as the one-shot detection task, *i.e.*, using the bounding box in the first frame as the only exemplar. By exploiting the domain specific information, Siamese-RPN surpasses the performance of the CF based methods. Some methods [24, 6, 38] attempt to improve the method [25] by using layer-wise and depthwise feature aggregations, simultaneously predicting target bounding box and class-agnostic binary segmentation, and using ellipse fitting to estimate the bounding box rotation angle and size for better performance. However, the aforementioned methods rely on the pre-set anchor boxes to regress the bounding box of target, which can not adapt to various motion patterns and scales of targets, especially when the *fast motion* and *occlusion* challenges occur.

To that end, in this paper, we present a siamese motion-

---

aware network (SiamMan) for visual tracking, which is formed by the siamese feature extraction subnetwork, followed by three paralleling branches, *i.e.*, the classification, regression, and localization branches. Similar to [25], the classification branch is used to distinguish the foreground from background, while the regression branch is used to regress the bounding box of target. To reduce the impact of manually designed anchor boxes to adapt to different motion patterns and scales of targets, we design a localization branch, which coarsely localizes the target to help the regression branch to generate more accurate results. Meanwhile, we introduce the global context module [5] into the localization branch to capture long-range dependency, which makes the tracker to be more robust to the large target displacement. In addition, we also design a multi-scale learnable attention module to guide these three branches to exploit discriminative features for better performance. The whole network is trained in an end-to-end fashion offline with the large-scale image pairs by the standard SGD algorithm with back-propagation [23] in the training sets of MS COCO [27], ImageNet DET/VID [32], and YouTube-BoundingBoxes [31] datasets. For inference, the visual object tracking is formulated as the local one-shot detection task by using the bounding box of target in the first frame as the only exemplar. Several experiments are conducted on five challenging benchmarks, *i.e.*, VOT2016 [20], VOT2018 [21], OTB2015 [39], UAV123 [29] and LTB35 [28]. Our SiamMan method sets a new state-of-the-art on four datasets, *i.e.*, VOT2016, VOT2018, OTB2015, and LTB35, and performs on par with the state-of-the-art on UAV123. Notably, it achieves 0.513 and 0.462 EAOs, improving 0.042 and 0.016 absolute values, *i.e.*, 8.9% and 3.6% relative improvements, compared to the second best tracker on VOT2016 and VOT2018. Moreover, ablation experiments are conducted to verify the effectiveness of different components in our method.

The main contributions of this work are summarized as follows.

- We propose a new siamese motion-aware network (SiamMan) for visual tracking, which is formed by a backbone feature extractor and three branches, *i.e.*, the classification, regression, and localization branches.
- To capture the long-range dependency, we integrate the global context module [5] into the localization branch, making the tracker to be more robust to large target displacement.
- We design a multi-scale learnable attention module to guide the network to exploit discriminative features for accurate results.
- SiamMan achieves the state-of-the-art results on four challenging benchmarks, *i.e.*, VOT2016, VOT2018, OTB2015, and LTB35, and performs on par with the state-of-the-art on UAV123.

2. Related work

Visual tracking aims to estimate the states, *i.e.*, sizes and locations, of target in the video sequence, with the given state in the first frame, which is an important and fundamental problems in computer vision community. Correlation filter based approach attracts much attention of researchers due to its computational efficiency and competitive performance [11, 9, 26]. In recent years, the focus of researchers shift to the deep neural network based methods, such as MDNet [30], ATOM [10], SINT [35], SiamFC [2], and SiamRPN [25]. MDNet [30] learn the share representation of targets from multiple annotated video sequences for tracking, which has separate branches of domain-specific layers for binary classification at the end of the network, and shares the common information captured from all sequences in the preceding layers for generic representation learning. ATOM [10] is formed by two components, *i.e.*, a target estimation module, and a target classification module. The target estimation module is trained offline to predict the intersection over union overlap between the target and an estimated bounding box, and the target classification module is learned online to provide high robustness against distractor objects in the scene.

Some other researchers attempt to use the Siamese network for visual tracking. SINT [35] and SiamFC [2] formulate the visual tracking problem as the pairwise similarity learning of the target in consecutive frames using the Siamese network. Dong *et al.* [12] use the triplet loss to train the Siamese network to exploit discriminative features instead of the pairwise loss, which can mine the potential relationship among exemplar, *i.e.*, positive and negative instances, and contains more elements for training. To take fully use of semantic information, He *et al.* [16] construct a twofold Siamese networks, which is composed of a semantic branch and an appearance branch, and each of them is a similarity-learning Siamese network. Abdelpakey and Shehat [1] use semantic and objectness information and produce a class-agnostic using a ridge regression network for object tracking.

After that, inspired by Region Proposal Network (RPN) in object detection, SiamRPN [25] formulates visual tracking as a local one-shot detection task in inference, which uses the Siamese network for feature extraction and RPN for target classification and regression. Fan and Ling [13] construct a cascaded RPN from deep high-level to shallow low-level layers in a Siamese network. Zhu *et al.* [43] design a distractor-aware Siamese networks for accurate long-term tracking by using an effective sampling strategy to control the distribution of training data, and make the model focus on the semantic distractors. SiamRPN++ [24] is improved from SiamRPN [25] by performing layer-wise and depth-wise aggregations, which not only improves the accuracy but also reduces the model size. Zhang and Peng
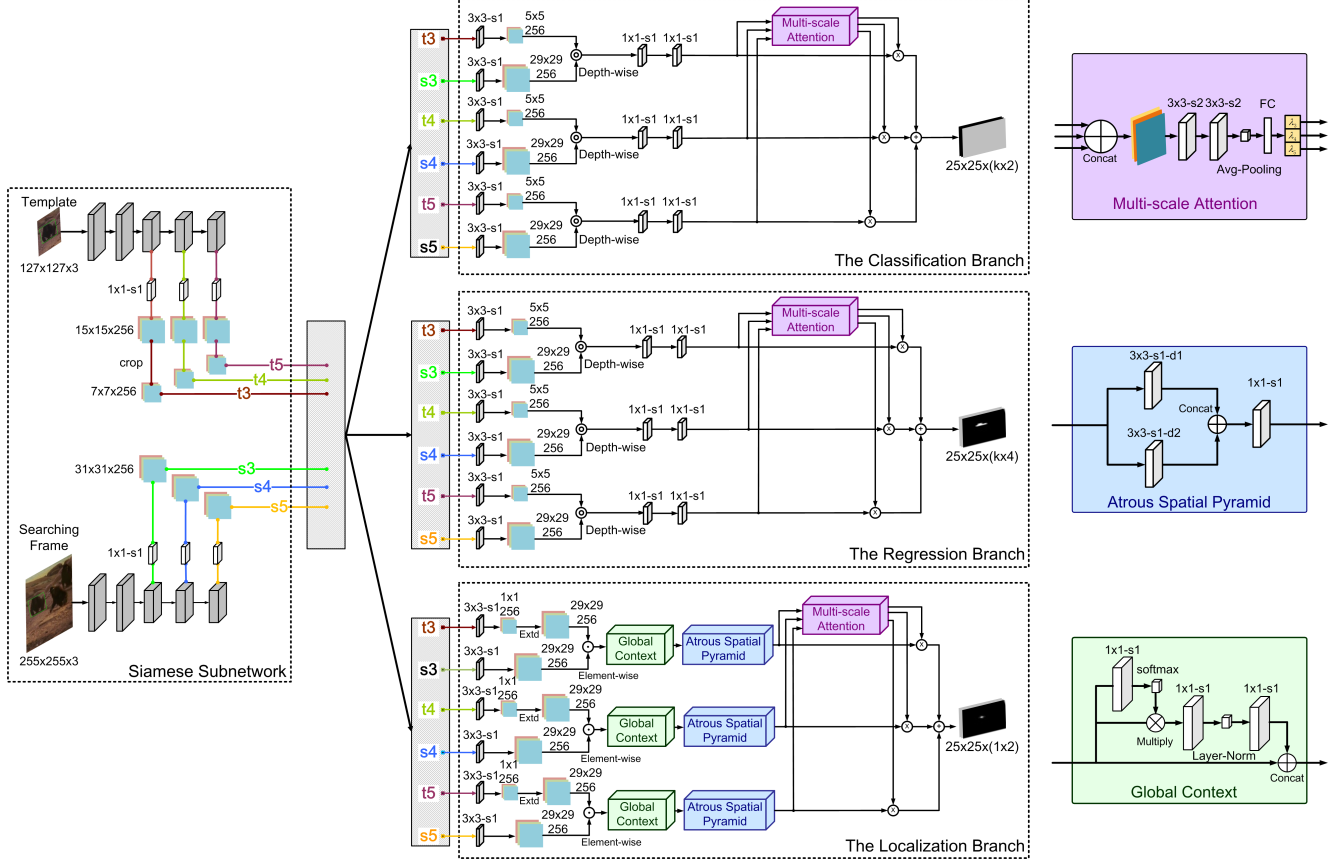
Figure 1. The architecture of our SiamMan method, which consists of the siamese feature extraction subnetwork followed by the classification, regression, and localization branches in parallel. The pairs of feature maps from different layers of the siamese feature extraction subnetwork are fed into the three branches. "3x3-s1-d2" denotes a convolution layer with 3×3 kernel, stride 1 and dilation rate 2. Best view in color.

[41] design a residual network for visual tracking with controlled receptive field size and network stride. Han *et al.* [15] introduce the anchor-free detection network into visual tracking directly. Moreover, SiamMask [38] combines the fully-convolutional Siamese tracker with a binary segmentation head for accurate tracking. To track the rotated target accurately, Chen *et al.* [6] improve SiamMask [38] by using ellipse fitting to estimate the bounding box rotation angle and size with the mask on the target. However, the aforementioned algorithms fail to consider the variations of target motion patterns, resulting in failures when the *fast motion*, *occlusion*, and *camera motion* challenges occur.

3. Siamese Motion-aware Network

As shown in Figure 1, our siamese motion-aware network is a feed-forward network, which is formed by a siamese feature extraction subnetwork, followed by three paralleling branches, *i.e.*, the classification branch, the regression branch, and the localization branch. The classification branch is designed to distinguish the foreground pro-

posals from the background, and the regression branch is used to regress the bounding box of target based on the pre-set anchor boxes. Inspired by [42], we integrate a localization branch, used to coarsely localize the target to help the regression branch to adapt to different motion patterns. Let k be the number of pre-set anchors. Thus, we have $2k$ channels for classification, $4k$ channels for regression and 2 channels for localization, and denote the output feature maps of the three branches as $\mathcal{O}^{cls}_{w \times h \times 2k}$, $\mathcal{O}^{reg}_{w \times h \times 4k}$, and $\mathcal{O}^{loc}_{w \times h \times 2}$. Notably, each point in $\mathcal{O}^{cls}_{w \times h \times 2k}$, $\mathcal{O}^{reg}_{w \times h \times 4k}$, and $\mathcal{O}^{loc}_{w \times h \times 2}$ contains $2k$, $4k$, and 2 channel vectors, representing the positive and negative activations of each anchor at the corresponding locations of original map for each branch. In the following sections, we will describe each module in our SiamMan in detail.

Siamese feature extraction subnetwork. Inspired by [25], we use the fully convolution network without padding is used in the Siamese feature extraction subnetwork. Specifically, there are two components in the Siamese feature extraction subnetwork, *i.e.*, the *template module* encoding the

3

target patch in the historical frame, and the *detection module* encoding the image patch including the target in the current frame. The two components share parameters in CNN. Let α be the target patch fed into the template module, and β be the image patch fed into the detection module. We denote the output feature maps of the Siamese feature extraction subnetwork at the i-th layer as $\phi_i(\alpha)$ and $\phi_i(\beta)$. Then, we split each of them into three branches, *i.e.*, $\phi_i^{\text{cls}}(\alpha)$ and $\phi_i^{\text{cls}}(\beta)$ for the classification branch, $\phi_i^{\text{reg}}(\alpha)$ and $\phi_i^{\text{reg}}(\beta)$ for the regression branch, and $\phi_i^{\text{loc}}(\alpha)$ and $\phi_i^{\text{loc}}(\beta)$ for the localization branch, by a convolution layer with kernel size 3×3 and stride 1, but keeping the number of channels unchanged. Similar to the previous work [24], we use the ResNet-50 network [17] as the backbone. To reduce the computational complexity, we extract the feature maps from the backbone with the channel 256 by one 1×1 convolutional layer, and then crop the center 7×7 regions [36] from the 15×15 feature maps as the template feature. Due to the paddings of all layers in the backbone, the 7×7 feature map can still represent the entire target region.

Classification branch. As shown in Figure 1, the classification branch takes the multi-scale features produced by the *template* and *detection* modules of the Siamese feature extraction subnetwork, *e.g.*, t3, s3, t4, s4, t5, and s5, to compute the correlation feature maps between the input *template* ($\phi_i^{\text{cls}}(\alpha)$) and *detection* ($\phi_i^{\text{cls}}(\beta)$) feature maps, *i.e.*,

$$F_{w \times h \times 2k}^{\text{cls}}(m) = \phi_m^{\text{cls}}(\alpha) \star \phi_m^{\text{cls}}(\beta), \tag{1}$$

where $\star$ denotes depth-wise convolution operation. We use two convolution layers with the kernel size 1×1 and stride size 1, to produce the features with $2k$ channels, *i.e.*, $\mathcal{F}_{w \times h \times 2k}^{\text{cls}}(m)$, $m = 1, \cdots, L$, where L is the total number of feature maps for prediction. After that, we use the multi-scale attention module to guide the branch to exploit discriminative features fore accurate results. Specifically, we first concatenate the feature maps at different layers, *i.e.*, $\mathcal{F}_{w \times h \times 2k}^{\text{cls}}(m)$, $m = 1, \cdots, L$, and use two convolutional layers with the kernel size 3×3 and stride size 2, followed by an average pooling and fully connected layers to produce the weights, *i.e.*, γ_i^{cls}. After that, the feature maps with different scales are summed with the weights γ_m^{cls} to generate the final predictions $\mathcal{O}_{w \times h \times 2k}^{\text{cls}}$, *i.e.*,

$$\mathcal{O}_{w \times h \times 2k}^{\text{cls}} = \sum_{m=1}^{L} \gamma_m^{\text{cls}} \cdot \mathcal{F}_{w \times h \times 2k}^{\text{cls}}(m). \tag{2}$$

Each point in the output $\mathcal{O}_{w \times h \times 2k}^{\text{cls}}$ is a $2k$ channel vector, indicating the positive and negative activations of each anchor at the corresponding locations of original map. Notably, the weights γ_m^{cls}, $m = 1, \cdots, L$, are learned in the training phase, *i.e.*, the gradients of the whole network can be back-propagated to update γ_m^{cls}, $m = 1, \cdots, L$. Please see Figure 1 for more details.

Regression branch. As described above, the regression branch is designed to generate the accurate bounding box of target in the current video frame. As shown in Figure 1, we compute the correlation feature maps between the input *template* and *detection* feature maps. For example, for the feature map at the m-th layer, the correlation feature map $F_{w \times h \times 4k}^{\text{reg}}(m)$ is computed as

$$F_{w \times h \times 4k}^{\text{reg}}(m) = \phi_m^{\text{reg}}(\alpha) \star \phi_m^{\text{reg}}(\beta), \tag{3}$$

where $\phi_m^{\text{reg}}(\alpha)$ and $\phi_m^{\text{reg}}(\beta)$ are the m-th feature map from the *template* and *detection* modules. After that, two convolution layers with the kernel size 1×1 and stride size 1, are applied on $F_{w \times h \times 4k}^{\text{reg}}(m)$ to produce the corresponding feature map $\mathcal{F}_{w \times h \times 4k}^{\text{reg}}(m)$, $m = 1, \cdots, L$, keeping the channel size $4k$ unchanged, where L is the total number of feature maps used for prediction. Similar to the *classification branch*, we use the multi-scale attention module with the learnable weights γ_m^{reg}, $m = 1, \cdots, L$, to make the branch focus on exploiting discriminative features to generate accurate results, *i.e.*,

$$\mathcal{O}_{w \times h \times 4k}^{\text{reg}} = \sum_{m=1}^{L} \gamma_m^{\text{reg}} \cdot \mathcal{F}_{w \times h \times 4k}^{\text{reg}}(m), \tag{4}$$

where $\mathcal{O}_{w \times h \times 4k}^{\text{reg}}$ is the output of the regression branch. Each point on $\mathcal{O}_{w \times h \times 4k}^{\text{reg}}$ contains a $4k$ channel vector, indicating the normalized distance between the predicted anchor box and the ground-truth bounding box.

Localization branch. In the visual tracking task, different targets have different motion patterns, *i.e.*, some targets move fast, while some targets move slowly. The *regression branch* relies on pre-set anchor boxes are inaccurate in challenging scenarios, such as *fast motion* and *small object*. To make our tracker adapt to various scales and motion patterns of targets, we introduce a *localization branch*, which is used to coarsely localizes the target to help the *regression branch* to produce accurate results. Specifically, taking the multi-scale features $\phi_i^{\text{loc}}(\alpha)$ and $\phi_i^{\text{loc}}(\beta)$ from the Siamese feature extraction subnetwork, we compute the correlation feature map as

$$F_{w \times h \times 2}^{\text{loc}}(m) = \mathbb{E}[\phi_m^{\text{loc}}(\alpha)] \odot \phi_m^{\text{loc}}(\beta), \tag{5}$$

where $\mathbb{E}[\cdot]$ denotes the resize operation to make the two feature maps to be the same size, and $\odot$ denotes element-wise multiplication operation, see Figure 1. After that, we insert the global context module [5] to integrate long-range dependency between target and background regions, making the tracker to be more robust to the large target displacement. Inspired by [7], we design the atrous spatial pyramid module to capture the context information at multiple scales, which applies two parallel atrous convolution with different rates, followed by a convolution layer with 1×1 kernel size and stride 1. In this way, we can generate the multi-scale

discriminative features $\mathcal{F}_{w \times h \times 2}^{\text{loc}}(m)$, where $m = 1, \cdots, L$. Then, similar to the *classification* and *regression* branches, we use the multi-scale attention module with the learnable weights γ_m^{loc}, $m = 1, \cdots, L$ to generate the final predictions. That is,

$$\mathcal{O}_{w \times h \times 2}^{\text{loc}} = \sum_{m=1}^{L} \gamma_m^{\text{loc}} \cdot \mathcal{F}_{w \times h \times 2}^{\text{loc}}(m). \quad (6)$$

Notably, each point on the prediction $\mathcal{O}_{w \times h \times 2}^{\text{loc}}$ is a two channel vector, representing the offset of the corresponding center location in original map.

Loss function. The loss function in our method is formed by three terms corresponding to three branches, *i.e.*, the classification loss L_{cls}, the regression loss L_{reg}, and the localization loss L_{loc}. The overall loss function is computed as:

$$\mathcal{L} = \lambda_{\text{cls}} L_{\text{cls}}(\text{u}, \text{u}^*) + \lambda_{\text{reg}} L_{\text{reg}}(\text{p}, \text{p}^*) + \lambda_{\text{loc}} L_{\text{loc}}(\text{c}, \text{c}^*), \quad (7)$$

where λ_{cls}, λ_{reg} and λ_{loc} are the parameters used to balance the three loss terms. u and u^* are the predicted and ground-truth labels of the target bounding boxes, p and p^* are the predicted and ground-truth bounding boxes, and c and c^* are the predicted and ground-truth labels of the center of target. We use the cross-entropy loss to supervise the *classification* and *localization* branches, and L1 loss to supervise for the *regression* branch.

Specifically, the classification loss $L_{\text{cls}}(\text{u}, \text{u}^*)$ is computed as

$$L_{\text{cls}}(\text{u}, u^*) = -\frac{1}{2} \sum_i \sum_j \sum_k \left(u_{i,j,k}^* \log u_{i,j,k} + (1 - u_{i,j,k}^*) \log(1 - u_{i,j,k}) \right), \quad (8)$$

where $u_{i,j,k}^*$ is the ground-truth label of the k-th anchor at (i, j) of the output $\mathcal{O}_{w \times h \times 2k}^{\text{cls}}$, and $u_{i,j,k}$ is the predicted label of the k-th anchor at (i, j) generated by the softmax operation from $\mathcal{O}_{w \times h \times 2k}^{\text{cls}}$ over 2 categories.

Meanwhile, we use the L1 loss function to compute the regression loss $L_{\text{reg}}(p, p^*)$, *i.e.*,

$$L_{\text{reg}}(\text{p}, \text{p}^*) = \frac{1}{N} \sum_i \sum_j \sum_k [u_{i,j,k}^* > 0] \|\delta(p_{i,j,k}, p_{i,j,k}^*)\|_1, \quad (9)$$

where N is the number of positive anchors, and the Iverson bracket indicator function $[u_{i,j,k}^* > 0]$ outputs 1 when the condition is true, *i.e.*, the anchor is not negative $u_{i,j,k}^* > 0$, and 0 otherwise. $p_{i,j,k} = (x, y, w, h)$ and $p_{i,j,k}^* = (x^*, y^*, w^*, h^*)$ are the predicted and ground-truth bounding boxes, where (x, y) and (x^*, y^*) are the center coordinates and (w, h) and (w^*, h^*) are the sizes. We use the normalized distances δ to compute the regression loss, *i.e.*, $\delta(p, p^*) = ((x^* - x)/x, (y^* - y)/y, \ln(w^*/w), \ln(h^*/h))$.

Moreover, we also use the cross-entropy loss for the localization branch as follows.

$$L_{\text{loc}}(\text{c}, \text{c}^*) = -\frac{1}{2} \sum_i \sum_j \left(c_{i,j}^* \log c_{i,j} + (1 - c_{i,j}^*) \log(1 - c_{i,j}) \right), \quad (10)$$

where $c_{i,j}^*$ is the ground-truth label of the center of target at (i, j) of the output $\mathcal{O}_{w \times h \times 2}^{\text{loc}}$, and $c_{i,j}$ is the predicted label of the center of target at (i, j) generated by the softmax operation from $\mathcal{O}_{w \times h \times 2}^{\text{loc}}$. Notably, we generate the ground-truth center location of the target c^* (where $c_{i,j}^* \in [0, 1]$) using the Gaussian kernel with the object size-adaptive standard deviation [22].

3.1. Training and Inference

Data augmentation. We use several data augmentation strategies such as blur, rescale, rotation, flipping and gray scaling to construct a robust model to adapt to variations of objects using the video sequences in MS COCO [27], ImageNet DET/VID [32], and YouTube-BoundingBoxes [31]. For the positive image pairs, we randomly select two frames from the same video sequences with the interval less than 100 frames, or different image patches including target object in the MS COCO and ImageNet DET datasets. Meanwhile, for the negative image pairs, we randomly select an image from the datasets and another one without including the same target. Notably, the ratio between the positive and negative pairs is set to $4 : 1$.

Anchors design and matching. For each point, we pave 5 anchors with stride 8 on each pixel, where the anchor ratios are set to $[1/3, 1/2, 1/1, 2/1, 3/1]$ and the anchor scale is set to 8. Meanwhile, during the training phase, we determine the correspondence between the anchors and ground-truth boxes based on the jaccard overlap. Specifically, if the overlap between the anchor and ground-truth box is larger than 0.6, the anchor is determined as positive. Meanwhile, if the overlap between the anchor and all ground-truth boxes is less than 0.3, the anchor is determined as negative.

Optimization. The whole network is trained in an end-to-end manner using the SGD optimization algorithm with 0.9 momentum and 0.0001 weight decay on the training sets of MS COCO [27], ImageNet DET/VID [32], and YouTube-BoundingBoxes [31] datasets. Notably, we use three stages to train the proposed method empirically. For the first two stages in the training process, we disable the multi-scale attention modules in the three branches, *i.e.*, set equal weights to different scales of features.

- In the first stage, the backbone ResNet-50 network in the siamese feature subnetwork is initialized by the pre-trained model on the ILSVRC CLS-LOC dataset [32]. We train the classification and regression branches in the first 10 epochs with other parameters fixed, and then train the siamese feature subnetwork, and the classification and regression branches in the next 10 epochs.
- In the second stage, we finetune the classification, regression and localization branches with other param-

eters fixed in the first 10 epochs, and then train the whole network in the next 10 epochs.

- In the third stage, we enable the multi-attention module and learn the weights of different scales of features with other parameters fixed in the first 15 epochs. After that, the whole network is finetuned in the next 5 epochs.

In each stage, we set the initial learning rate to 0.001, and gradually increase it to 0.005 in the first 5 epochs. We decrease it to 0.0005 in the next 15 epochs.

Inference. In the inference phase, our tracker takes the current video frame and the template target patch as input, and outputs the classification, regression, and localization results. Then, we perform softmax operation on both the outputs of the classification and localization results to obtain the positive activations, *i.e.*, u with the size $w \times h \times k$, and c with the size $w \times h \times 1$. After that, we expand the localization result c to make it to the same size of the classification result u. In this way, the final prediction is computed by the weight combination of four terms, *i.e.*, the localization result c, the classification result u, the cosine window ξ with the size $w \times h$ (expanding to $w \times h \times k$), and the scale change penalty ρ with the size $w \times h \times k$ [25],

$$\Theta_{w \times h \times k} = \omega_2 \cdot \rho \cdot \left(\omega_1 \cdot u + (1 - \omega_1) \cdot c \right) + (1 - \omega_2) \cdot \xi, \tag{11}$$

Notably, The cosine window ξ is used to suppress the boundary outliers [19], and the scale change penalty ρ to suppress large change in size and ratio [25]. The weights ω_1 and ω_2 are used to balance the above terms, which are set to 0.7 and 0.6, empirically. After that, we can obtain the optimal center location and scale of target based on the maximal score on $\Theta_{w \times h \times k}$. Notably, the target size is updated by the linear interpolation to guarantee the smoothness of size.

4. Experiment

Our SiamMan method is implemented using the Pytorch tracking platform PySOT[1]. Several experiments are conducted on five challenging datasets, *i.e.*, VOT2016 [20], VOT2018 [21], OTB100 [39], UAV123 [29] and LTB35 [28], to demonstrate the effectiveness of the proposed method. All experiments are conducted on a workstation with the Intel i7-7800X CPU, 8G memory, and 2 NVIDIA RTX2080 GPUs. The average tracking speed is 45 fps. The source code and models will be released after the paper is accepted.

Evaluation protocol. For the VOT2016 [20] and VOT2018 [21] datasets, we use the evaluation protocol in the VOT

Challenge [20, 21], *i.e.*, the *Expected Average Overlap* (EAO), *Accuracy* (A), and *Robustness* (R) are used to evaluate the performance of trackers. The *Accuracy* score indicates the average overlap of the successfully tracked frames, and the *Robustness* score indicates the failure rate of the tracking frames[2]. EAO takes both accuracy and robustness into account, which is used as the primary metric for ranking trackers.

Meanwhile, for the OTB100 [39] and UAV123 [29] datasets, we use the success and precision scores to evaluate the performance of trackers based on the evaluation methodology in [39]. The success score is defined as the area under the success plot, *i.e.*, the percentage of successfully tracked frames[3] *vs.* bounding box overlap threshold in the interval $[0, 1]$. The precision score is computed as the percentage of frames whose predicted location is within a given distance threshold from the center of ground-truth box based on the Euclidean distance on the image plane. We set the distance threshold to 20 pixels in our evaluation. In general, the success score is used as the primary metric for ranking trackers.

For the long-term tracking LTB35 dataset [28], we use three metrics including tracking precision (P), tracking recall (R) and tracking F-score in evaluation. The tracking methods are ranked by the maximum F-score over different confidence thresholds, *i.e.*, $F = \frac{2P \cdot R}{P + R}$.

4.1. State-of-the-art Comparison

We compare the proposed method to the state-of-the-art trackers on five challenging datasets. For a fair comparison, the tracking results of other trackers are directly taken from the published papers.

VOT2016. We conduct experiments on the VOT2016 dataset [20] to evaluate the performance of our SiamMan method in Table 1. VOT2016 contains 60 sequences. Each sequence is per-frame annotated with the following visual attributes: *occlusion*, *illumination change*, *motion change*, *size change*, *camera motion*, and *unassigned*. As shown in Table 1, our SiamMan method achieves the best EAO score 0.513 and the second best robustness score 0.149. Notably, our method sets a new state-of-the-art by improving 0.042 absolute value, *i.e.*, 8.9% relative improvement, compared to the second best tracker PTS [37]. However, our method produce a relative lower accuracy score compared to SiamMask [38] and SiamMask_E [6]. The SiamMask and SiamMask_E methods estimate the rotated bounding box based on the mask generated by the segmentation head, resulting in relative more accurate bounding box, especially

---

[2]We define the failure of tracking if the overlap between the tracking result and ground-truth is reduced to 0.

[3]If the overlap between the predicted bounding box and ground-truth box in a frame is larger than a threshold, we regard the frame as a successfully tracked frames.

Table 1. Comparisons with the state-of-the-art on VOT2016 [20] in terms of EAO, robustness, and accuracy. ∗ denotes that the result is obtained using the PySOT platform.

Tracker	Accuracy	Robustness	EAO
C-COT [11]	0.539	0.238	0.331
SiamRPN [25]	0.560	1.080	0.344
FCAF [15]	0.581	1.020	0.356
C-RPN [13]	0.594	0.950	0.363
SiamRPN+ [41]	0.580	0.240	0.370
ECO [9]	0.550	0.200	0.375
ASRCF [8]	0.563	0.187	0.391
DaSiamRPN [43]	0.610	0.220	0.411
SiamMask∗ [38]	**0.643**	0.219	0.455
SiamRPN++∗ [24]	0.642	0.196	0.464
SiamMask_E∗ [6]	**0.677**	0.224	0.466
PTS [37]	0.642	**0.144**	**0.471**
SiamMan∗	0.636	**0.149**	**0.513**

Table 2. Comparison results on VOT2018 [21]. ∗ denotes that the result is obtained using the PySOT platform.

Tracker	Accuracy	Robustness	EAO
SiamFC [2]	0.503	0.585	0.188
DSiam [14]	0.215	0.646	0.196
SiamRPN [25]	0.490	0.460	0.244
ECO [9]	0.484	0.276	0.280
SA_Siam_R [16]	0.566	0.258	0.337
DeepSTRCF [26]	0.523	0.215	0.345
DRT [34]	0.519	0.201	0.356
RCO [21]	0.507	0.155	0.376
UPDT [4]	0.536	0.184	0.378
DaSiamRPN [43]	0.586	0.276	0.383
MFT [21]	0.505	**0.140**	0.385
LADCF [40]	0.503	0.159	0.389
DomainSiam [1]	0.593	0.221	0.396
PTS [37]	0.612	0.220	0.397
ATOM [10]	0.590	0.204	0.401
SiamRPN++∗ [24]	0.601	0.234	0.415
SiamMask∗ [38]	**0.615**	0.248	0.423
DiMP-50 [3]	0.597	**0.153**	0.440
SiamMask_E∗ [6]	**0.655**	0.253	**0.446**
SiamMan∗	0.605	0.183	**0.462**

for the non-rigid targets. Compared to SiamRPN++ [24], our SiamMan method produces 0.049 and 0.005 higher EAO and robustness scores, indicating that the localization branch can significantly decrease the tracking failure.

VOT2018. The VOT2018 dataset consists of 60 challenging video sequences, which is annotated with the same standard as VOT2016 [20]. We evaluate the proposed SiamMan method on VOT2018 [21], and report the results in Table 2. As shown in Table 2, our SiamMan method outperforms the state-of-the-art methods in terms of the primary ranking metric EAO. SiamMask_E [6] and SiamMask [38] estimate the rotated bounding boxes of targets based on the segmentation results, producing higher accuracy scores, i.e., 0.655
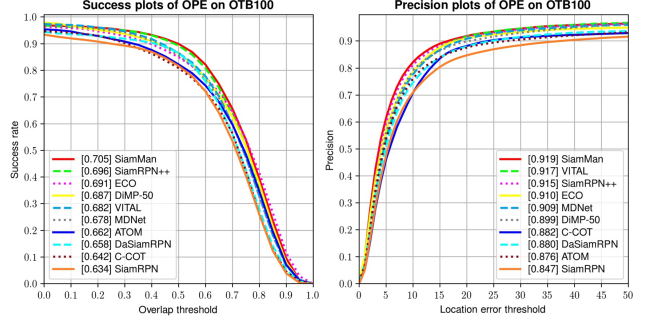

Figure 2. Success and precision plots on the OTB100 dataset [39].

and 0.615. Our SiamMan method outperforms SiamRPN++ [24], i.e., improving 0.047 (0.462 vs. 0.415) EAO and 0.051 (0.183 vs. 0.234) robustness, which fully demonstrates the effectiveness of the designed localization branch and multi-scale attention module.

OTB100. OTB100 [39] is a challenging dataset, which consists of 100 video sequences. We compare our SiamMan method with several representative trackers, i.e., SiamRPN++ [24], ECO [9], DiMP-50 [3], VITAL [33], MDNet [30], ATOM [10], DaSiamRPN [43], C-COT [11], and SiamRPN [25], shown in Figure 2. As shown in Figure 2, our method achieves the best performance in both success and precision scores, i.e., 0.705 success score and 0.919 precision score. VITAL [33] achieves the second best precision score 0.917 but much worse success score 0.682. Compared to SiamRPN++ [24], our method improves 0.009 in success score (i.e., 0.705 vs. 0.696) and 0.004 in precision score (0.919 vs. 0.915).

UAV123. We also evaluate our SiamMan method on the UAV123 dataset [29] in Figure 3. The dataset is collected from an aerial viewpoint, which includes 123 sequences in total with more than 110, 000 frames. As shown in Figure 3, our method performs on par with the state-of-the-art tracker DiMP-50 [3], i.e., it produces the same success score 0.648 but a little bit worse precision score (0.857 vs. 0.858). Compared to SiamRPN++ [24], our method achieves higher success (0.648 vs. 0.642) and precision scores (0.857 vs. 0.840). It is attributed to the localization branch and the multi-scale attention module introduced in our tracker.

LTB35. In addition, we also evaluate our SiamMan tracker on the LTB35 dataset [28], which is first presented in the VOT2018-LT challenge [21]. It includes 35 sequences with 14, 687 frames. Each sequence contains 12 long-term target disappearing cases in average. We compare the proposed SiamMan method to several best-performing methods in the VOT2018-LT challenge [21] and SiamRPN++ [24] in Figure 4. As shown in Figure 4, our method performs better than all those methods. Specifically, we achieve 64.1%
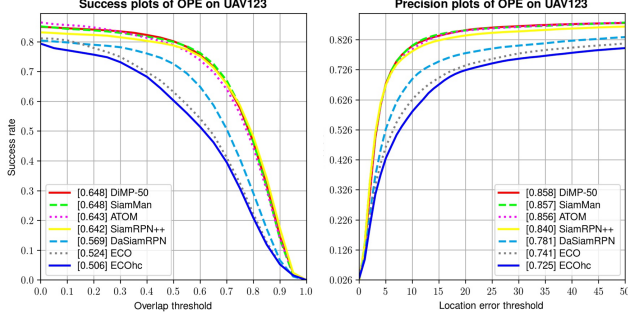
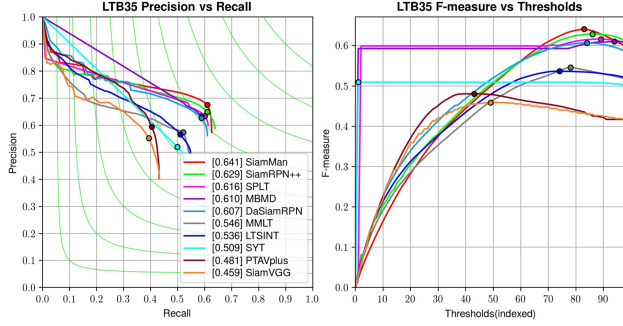Figure 3. Success and precision plots on the UAV123 dataset [29].

Figure 4. Evaluation results on the LTB35 dataset [28], including recall and precision (left) and F-score (right).

F-score, *i.e.*, 1.2% higher than the second best method SiamRPN++ [24] (0.641 *vs.* 0.629). The results indicate that our method using the localization branch and multi-scale attention module performs well in long-term tracking even without using the re-detection module.

4.2. Ablation Study

To validate the effectiveness of different components, *i.e.*, the localization branch, the global context module, and the multi-scale attention module, in our method, we conduct several ablation experiments on the challenging VOT2016 [20] and VOT2018 [21] datasets. Notably, we use the same parameter settings and training data for a fair comparison. In addition, we also analyze the performance of the Siamese network based trackers in 11 attributes on the OTB100 dataset [39] to further demonstrate the effectiveness of the proposed method.

Localization branch. If we remove the localization branch, the global context module, and the multi-scale attention module, our SiamMan degenerates to the original SiamRPN++ method [24]. After integrating the localization branch, EAO is improved from 0.464 to 0.488 on VOT2016 and 0.415 to 0.432 on VOT2018, respectively. This significant improvements demonstrate that the localization branch is critical for the tracking performance.

Table 3. Effectiveness of different components in the proposed method based on EAO.

Component	SiamMan			
localization branch?	✓	✓	✓	✓
global context?		✓		✓
multi-scale attention?			✓	✓
VOT2016	0.464	0.488	0.494	0.504	**0.513**
VOT2018	0.415	0.432	0.446	0.447	**0.462**

Global context module. In addition, we use global context module in the localization branch to capture long-range dependency. To demonstrate the effectiveness of the global context model, we construct a variant, *i.e.*, remove the global context module from our SiamMan tracker, and evaluate it on VOT2016 and VOT2018, shown in the fourth column in Table 3. As shown in the fourth and fifth columns in Table 3, if we remove the global context module, the EAO scores drop 0.009 (0.513 *vs.* 0.504) and 0.015 (0.462 *vs.* 0.447), respectively. The results indicate that the global context module in the localization branch noticeably improve the tracking accuracy.

Multi-scale attention. Furthermore, to validate the effectiveness of the multi-scale attention module, we construct a variant, *i.e.*, remove the multi-scale attention module from our SiamMan tracker, and evaluate it on VOT2016 and VOT2018, shown in the third columns in Table 3. As shown in Table 3, we find that the multi-scale attention module significantly improves the performance of the proposed tracker on both VOT2016 and VOT2018, *i.e.*, improving 0.019 (0.513 *vs.* 0.494) and 0.016 (0.462 *vs.* 0.446) EAOs. The learnable multi-scale attention module constructs an optimal combinations of multi-scale features from the siamese feature extraction subnetwork, which is effective to guide the three branches, *i.e.*, the classification, regression, and localization branches, to exploit discriminative features better performance.

Performance on different attributes. To verify the effectiveness of our method in detail, we also report the success score of the different trackers on different attributes in Figure 5. Compared to the Siamese network based trackers, *i.e.*, SiamRPN++ [24], SiamRPN [25], C-RPN [13] and DaSiamRPN [43], and other state-of-the-art methods, *i.e.*, DiMP-50 [3] and ATOM [10], our method performs the best in the most of the attributes, especially in *fast motion*, *out-of-view*, *low resolution* and *background clutters*. Most of the previous siamese network based trackers rely on the pre-set anchor boxes, making it difficult to adapt to different motion patterns and scales of targets, resulting in inaccurate tracking results in challenging scenarios such as *fast motion* or *low resolution* (*i.e.*, indicating small scale target). The localization branch in the proposed method is effective to
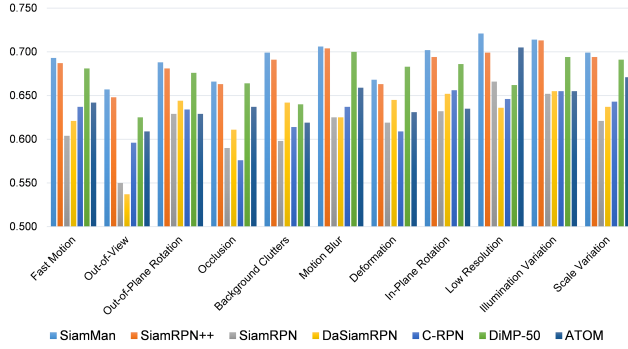
Figure 5. Success score of the proposed method in each attribute on OTB100 [39].

coarsely localize the target to help the regression branch to generate accurate results, making our tracker to be less sensitive to the variations of motion patterns and scales with the preset anchors.

5. Conclusion

In this paper, we propose a novel siamese motion-aware network for visual tracking, which integrates a new designed localization branch to deal with various motion patterns in complex scenarios. It coarsely localizes the target to help the regression branch to generate more accurate results, and lead to less tracking failures, especially when the fast motion, occlusion, and low resolution challenges occur. Moreover, we design a multi-scale attention module to guide these three branches to exploit discriminative features for better performance. Our tracker sets the new state-of-the-art on four challenging tracking datasets, *i.e.*, VOT2016, VOT2018, OTB2015, and LTB35, and performs on par with the state-of-the-art on UAV123, with the real-time running speed 45 frame-per-second.

References

[1] Mohamed H. Abdelpakey and Mohamed S. Shehata. Domainsiam: Domain-aware siamese network for visual object tracking. *CoRR*, abs/1908.07905, 2019.

[2] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, pages 850–865, 2016.

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019.

[4] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, pages 493–509, 2018.

[5] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *CoRR*, abs/1904.11492, 2019.

[6] Bao Xin Chen and John K. Tsotsos. Fast visual object tracking with rotated bounding boxes. *CoRR*, abs/1907.03892, 2019.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2018.

[8] Kenan Dai, Dong Wang, Huchuan Lu, Chong Sun, and Jianhua Li. Visual tracking via adaptive spatially-regularized correlation filters. In *CVPR*, pages 4670–4679, 2019.

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *CVPR*, pages 6931–6939, 2017.

[10] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: accurate tracking by overlap maximization. In *CVPR*, 2019.

[11] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, pages 472–488, 2016.

[12] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *ECCV*, pages 472–488, 2018.

[13] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019.

[14] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, pages 1781–1789, 2017.

[15] Guang Han, Hua Du, Jixin Liu, Ning Sun, and Xiaofei Li. Fully conventional anchor-free siamese networks for object tracking. *Access*, 7:123934–123943, 2019.

[16] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, pages 4834–4843, 2018.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 FPS with deep regression networks. In *ECCV*, pages 749–765, 2016.

[19] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015.

[20] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman P. Pflugfelder, Luka Cehovin, Tomás Vojír, Gustav Häger, Alan Lukezic, Gustavo Fernández, and *et al.* The visual object tracking VOT2016 challenge results. In *ECCVW*, pages 777–823, 2016.

[21] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman P. Pflugfelder, Luka Cehovin Zajc, Tomás Vojír, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, Gustavo Fernández, and *et al.* The sixth visual object tracking VOT2018 challenge results. In *ECCVW*, pages 3–53, 2018.

[22] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 765–781, 2018.

[23] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and

Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[24] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[25] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, pages 8971–8980, 2018.

[26] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming-Hsuan Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *CVPR*, pages 4904–4913, 2018.

[27] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755, 2014.

[28] Alan Lukezic, Luka Cehovin Zajc, Tomás Vojír, Jiri Matas, and Matej Kristan. Now you see me: evaluating performance in long-term visual tracking. *CoRR*, abs/1804.07056, 2018.

[29] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *ECCV*, pages 445–461, 2016.

[30] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016.

[31] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, pages 7464–7473, 2017.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[33] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W. H. Lau, and Ming-Hsuan Yang. VITAL: visual tracking via adversarial learning. In *CVPR*, pages 8990–8999, 2018.

[34] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, pages 489–497, 2018.

[35] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, pages 1420–1429, 2016.

[36] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, pages 5000–5008, 2017.

[37] Jianren Wang, Yihui He, Xiaobo Wang, Xinjia Yu, and Xia Chen. Prediction-tracking-segmentation. *CoRR*, abs/1904.03280, 2019.

[38] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019.

[39] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015.

[40] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. *TIP*, 28(11):5596–5609, 2019.

[41] Zhipeng Zhang, Houwen Peng, and Qiang Wang. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, 2019.

[42] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019.

[43] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, pages 103–119, 2018.