Visual Object Tracking based on Adaptive Siamese and Motion Estimation Network

Hossein Kashiani*, Shahriar B. Shokouhi

School of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran

Abstract

Recently, convolutional neural network (CNN) has attracted much attention in different areas of computer vision, due to its powerful abstract feature representation. Visual object tracking is one of the interesting and important areas in computer vision that achieves remarkable improvements in recent years. In this work, we aim to improve both the motion and observation models in visual object tracking by leveraging representation power of CNNs. To this end, a motion estimation network (named MEN) is utilized to seek the most likely locations of the target and prepare a further clue in addition to the previous target position. Hence the motion estimation would be enhanced by generating a small number of candidates near two plausible positions. The generated candidates are then fed into a trained Siamese network to detect the most probable candidate. Each candidate is compared to an adaptable buffer, which is updated under a predefined condition. To take into account the target appearance changes, a weighting CNN (called WCNN) adaptively assigns weights to the final similarity scores of the Siamese network using sequence-specific information. Evaluation results on well-known benchmark datasets (OTB100, OTB50 and OTB2013) prove that the proposed tracker outperforms the state-of-the-art competitors.

Keywords: Object Tracking, Siamese Network, Convolutional Neural Networks, Motion Estimation.

*Corresponding author

Email addresses: hossein_kashiyani@alumni.iust.ac.ir (Hossein Kashiani), bshokouhi@iust.ac.ir (Shahriar B. Shokouhi)

1. Introduction

Visual object tracking is the process of determining the target location in a video sequence. It is one of the many remarkable research topics in computer vision that has gained considerable attention over the past decade. It can be applied to numerous applications, including video indexing, activity recognition, augmented reality, vehicle navigation and so forth. Noticeable efforts have been made to improve object tracking accuracy in the past decade; however, it is still an open and active research area since different variations have been imposed on the object of interest during the tracking process.

Recently, the success of convolution neural networks (CNNs) has drawn much attention to various areas in computer vision, like image recognition [1], object detection [2, 3] and object tracking [4, 5, 6, 7]. CNNs can exploit a wider range of features from an arbitrary target in contrast to low-level hand-crafted features. Low-level convolutional features in earlier CNN layers contain higher spatial resolution appropriate for accurate object localization. While high-level convolutional features in later layers retain a more semantic context [8] helpful for distinguishing the foreground object from the background.

Generally, a tracking system consists of two fundamental components: 1) An observation model, which models the appearance of the target and validates candidates over time; 2) A motion model, which generates the number of candidates over time [9, 10].

In this paper, we aim to enhance the performance of both these models simultaneously. In spite of the substantial amount of researches in CNN-based trackers, there are limited numbers of studies in augmenting motion model [11, 12]. A naive motion model, which is conventionally employed in tracking algorithms [13, 14, 15, 16], presumes that the state transition of the target can be modeled by a Gaussian distribution. Consequently, new candidates are generated centered at the former target state by employing a Gaussian distribution. To cope with fast and complex motion, the trackers using this motion model tend to drift

away from the target. Furthermore, if the previous estimated location entails background distractions, new candidates are generated based on the previous incorrect target state, and hence the tracker will deviate.

To address the aforementioned issues, motivated by [13, 12, 17, 18], a motion estimation network (named MEN) is utilized to prepare another evidence of target location in addition to the output of the Siamese network. Therefore, motion model would be improved by generating a small number of candidates, but near the two most likely locations (the previous target location and the location detected by the MEN). In order to take advantage of sequence-specific cues, the last two convolutional layers of the MEN are updated over time and by this way at each time-step the MEN takes into account the information of current and previous frames.

Based on the observation model, CNN-based visual object tracking is approximately divided into discriminative and generative approaches [19]. Discriminative methods consider object tracking as binary classification and differentiate the target from the background candidates using either a new CNN model or a pretrained CNN model [1, 20]. In order to consider the object appearance variations, these methods can be integrated into an online fine-tuning strategy [13, 21]. In the discriminative trackers, online fine-tuning during tracking process can enhance the overall performance of the tracker at the expense of increased computational cost. On the other hand, generative methods seek to learn matching function to make a comparison between the target template and calculated candidates using CNN while maintaining high-speed performance [6, 7, 22].

Siamese network is one of the prominent models in generative methods which achieves promising accuracy as well as high speed. A Y-shaped Siamese network aims to learn the similarity rate between the initial target and the sampled candidates while remaining unchanged over time [7, 22]. Exploiting one fixed matching function impedes the Siamese-based tracker to take sequence-specific information into account, whereas discriminative trackers take this information into consideration [21]. Thus, utilizing nonadjustable matching function and

bypassing online updating would make Siamese-based trackers less adaptable in dealing with target appearance changes, background distractions and coexisting of confusing objects. To address these issues, sequence-specific information can be integrated into the Siamese-based model trained offline. In this paper, to reach this objective, a particular Siamese network is followed by a weighting CNN (called *WCNN*) to leverage sequence-specific information. Figure 1 represents the proposed framework and illustrates the collaboration of the WCNN and the Siamese network. The WCNN is updated over time to consider target appearance variations. The influence of sequence-specific information is exerted over the final similarity score of the Siamese network using a weighting mechanism. Integrating sequence-specific information with the Siamese network is worthwhile not only to discern the most similar candidates to target but also to exploit sequence-specific cues in weighting the candidates.

To sum up, the proposed tracker represents a step forward in making the tracker robust against a wide range of challenging scenarios. Evaluations on benchmark datasets (OTB100 [23], OTB50 and OTB2013[24]) demonstrate the superiority of the proposed tracker over the state-of-the-art competitors.

2. Related Work

In this section, two main categories of CNN-based trackers related to our method are presented.

2.1. Discriminative CNN-based Trackers

With the advent of CNN and its notable success, researchers have tended to take advantage of CNN power in visual object tracking. Nam and Han [13] try to leverage a shallow CNN, which is pretrained on a tracking-specific dataset. The pretrained CNN is composed of a shared and sequence-specific part. The shared section extracts the generic feature representation of each sequence. While sequence-specific section differentiates the target from the background in each sequence. Authors in [13] try to improve the accuracy of the tracker by updating

sequence-specific section at the cost of decreased tracker speed. This time-consuming procedure of online updating is alleviated in [25]. Park et al.[25] exploit an offline meta-learning approach to tune the initial state of sequence-specific section to be updated faster in online tracking. Qi et al.[26] hedge deep features from six CNN layers and construct six weak trackers based on correlation filter. Then, these weak trackers are fused into a strong one.

2.2. Generative CNN-based Trackers

Generative CNN-based trackers aim to find the most resembling candidate to the target via a generic matching function. One of the outstanding networks in generative approaches undoubtedly pertains to the Siamese networks, which are increasingly being used in visual object tracking. Bertinetto et al.[7] adopt a similarity learning framework with no update strategy to train a fully-convolutional Siamese network and find the target within a large area. Two branches of the Siamese network is linked together via a cross-correlation layer. After training the network, a scalar-valued score map specifies the location of the target. In [27], the target image and a large search area are transmitted to the inputs of a Y-shaped network and, consequently, the network outputs a binary response map indicating the location of the target. Unlike the network architecture in [7], Chen et al.[27] concatenate the last three fully connected (fc) layers to capture spatiotemporal features [28]. Tao et al.[22] prefer to supplant the concatenation layers with a normalization layer before the loss layer in comparison with the other approaches [7, 27]. By this way, authors in [22] attempt to sustain the direction of the feature, while embedding them on a unit sphere. He et al.[29] construct twofold Siamese networks based on [7] architecture. These two networks are trained separately to keep their heterogeneity and they are only connected when their similarity scores are calculated. Zhang et al.[30] propose structured Siamese network to not only account local patterns of the target but also consider their structural relationships. They utilize a local pattern detector to automatically accomplish this task. Despite the successes of the aforementioned trackers, most of them can not satisfactorily cope

with large object appearance variations due to the parameters sharing in the Siamese network [28]. In this paper, a weighting strategy is incorporated into the Siamese network to make the tracker more versatile in dealing with large object appearance variations. It is worth noting that our proposed method bears some similarity to a recent paper [31] as they incorporate pretrained Siamese network into sequence-specific updating. However, our approach differs from [31] in several respects. First, Our proposed Siamese network is based on the SINT [22] architecture that integrates hierarchical features in order to consider both semantic and spatial details. In addition, to efficiently emphasize the fine-to-coarse features quality rather than features scale, the normalization layer is used in the SINT architecture to make features restricted to a unit sphere. While the authors of [31] do not capture fine-to-coarse spatial features. Second, despite fine-tuning the generic Siamese network in [31] to adapt appearance variations, the candidates are solely evaluated by their distances from the ground truth. While, we utilize an adaptable buffer, which is updated by the best candidates over time to account target appearance changes during tracking process. Third, [31] incorporate sequence-specific information into Siamese tracker with fine-tuning the last three layers of the network and by this way they do not increase computational complexity at the cost of losing abstract representation of Siamese network. Compared with [31], we use a pretrained network WCNN fine-tuned over time for weighting candidates according to their location-specific information. This network is constituted by two 1×1 convolutional layers to reduce the trainable parameters and also preclude network over-fitting.

3. The proposed Tracker

In this section, our proposed method is thoroughly explained. First, MEN is described in depth and then the contribution of baseline Siamese network and its details are discussed. Lastly, the strategies of inference and updating are also presented. The whole framework of our proposed method is illustrated in Figure 1.

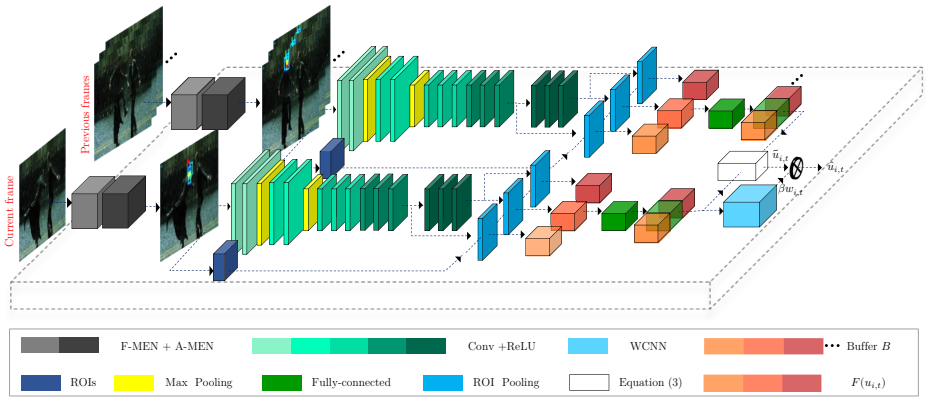


Figure 1: Online tracking framework at time step t. This network consists of three components: 1) MEN, which enhances the sampling procedure by providing a further cue (red point) for sampling candidates in addition to the previous target position (yellow point); 2) Siamese network constructed from two identical branches to measure the similarity of its inputs; 3) WCNN, which down-weights distracting candidates using sequence-specific information. The proposed tracking procedure at time step $t(> 1)$ is included in Algorithm 1.

3.1. Motion Estimation Network

Motion estimation network aims to enhance the performance of the motion model in complex and fast target movements. A conventional approach used in motion model tries to propagate new candidates according to the previous estimated target location using Gaussian distribution. In such approach, it is assumed that the distance of the target from the ground truth does not exceed a predefined threshold. Hence, when a tracker drifts away in complex movements or challenging scenarios, there is not much chance to redetect the target in the following frames. In order to equip our tracker with the redetection scheme, MEN (inspired by [13, 12]) is utilized. As shown in Figure 1, this network consists of an adaptive and a fixed component, named *F-MEN* and *A-MEN*. The F-MEN pretrained by VGG-M [32] captures the spatial information of an area at the center of the previous target location. This area is resized to $107 \times 107 \times 3$ before entering into the network. The F-MEN is made up of the first convolutional layer of VGG-M which is followed by a Rectified Linear Unit (ReLU) and a Local Response Normalization (LRN). The F-MEN outputs a $51 \times 51 \times 96$ feature. Compared with the frozen F-MEN, the A-MEN tries to

leverage the sequence-specific details. Namely, it attempts to adaptively deal with the target appearance variations while benefiting from both the temporal and long-term information of the target in each sequence simultaneously. To this end, it needs to be updated over time. A-MEN consists of two 1×1 convolutional layers trained in the first frame. To train A-MEN, the score map (with size of $51 \times 51 \times 2$) is fed into a softmax layer, which discriminates positive candidates from negative ones. The loss of the score map is calculated as follows:

$$L(y, z) = -\frac{1}{|B|} \sum_{s \in B} y[s] \log(\frac{\exp(z[s])}{\sum_{j=1}^{2} \exp(z[s]_j)}) \tag{1}$$

where $z : B \to \mathbb{R}$ represents the real-valued score map and $y[s] \in \{2, 1\}$ denotes the ground-truth label in each location $s \in B$ in the score map.

As [7, 12], the positive and negative candidates in the score map are determined by:

$$y[s] = \begin{cases} 2, & \text{if } \|o - o_g\| \leq R \\ 1, & \text{otherwise,} \end{cases} \tag{2}$$

where o_g and R stand for the center of the previous predicted location and a predefined radius that separates positive candidates from negative ones. On the score map, positive candidates are located within the radius R of the o_g. To remove class imbalance, the losses are also weighted according to class cardinality, i.e., $\{\frac{y[s]}{2\sum_i y[i]}\}_{y[s]\in\{2,1\}}$.

After training the A-MEN, the score map of the object of interest is back-propagated to the input image in order to find the most probable region. Figure 2 depicts the results of the back-projection for two sequences. Consequently, new candidates can be generated employing Gaussian distribution at the center of two points: 1) The location in the original input image corresponding to the maximum point in the score map of the MEN. This location is shown with a yellow cross in Figure 2; 2) The center of the ground truth bounding box (from the second frame on, this location is obtained employing the Siamese network).

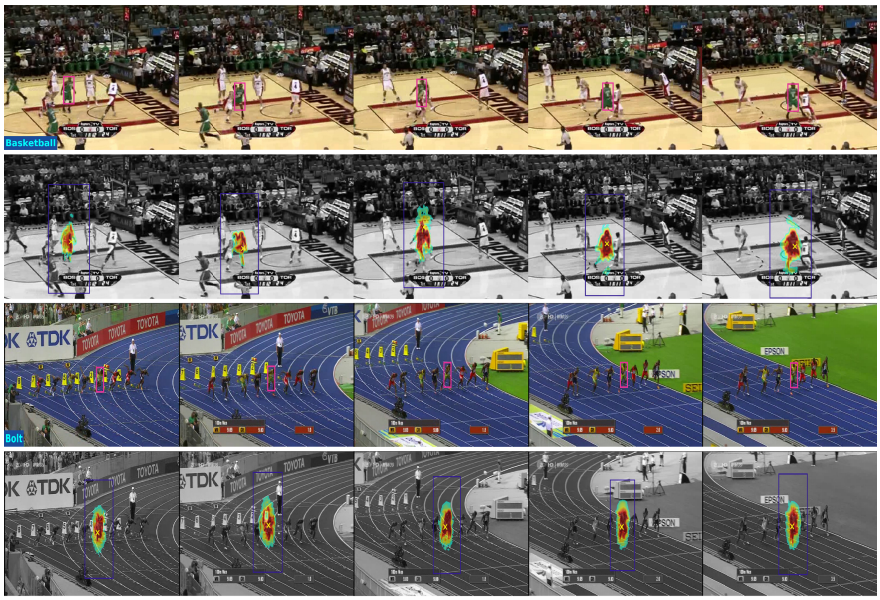


Figure 2: In the even rows, the heat maps are obtained by mapping score maps of the MEN to the input frames (the odd rows). The ground truth bounding boxes and the search regions are represented by pink and blue colors in odd and even rows, respectively. The blue boxes are about four times greater than the size of the corresponding pink boxes. The yellow cross corresponds to the maximum value in the score map. Best viewed in color and magnification.

Equipped with these two plausible locations, we can presume that candidates are subject to a Gaussian distribution around the locations. Let $u_t = (l_x, l_y, s)$ denote the state variable, where l_x, l_y and s indicate the center coordinates and the scale of the target bounding box, respectively. Then, the target candidates are drawn using Gaussian distribution as $\mathcal{N}(u_t, \hat{u}_t^m, \Sigma)$ around the first location and $\mathcal{N}(u_t, \hat{u}_{t-1}^f, \Sigma)$ around the second location, where Σ denotes a diagonal covariance matrix. Afterwards, the total generated candidates are fed into the Siamese network to determine which candidate resembles the target more closely.

3.2. Baseline Siamese Network

Siamese network has recently drawn much attention as an approach of similarity learning in visual object tracking. The convolutional architecture of our proposed method for similarity learning is constructed based on the Siamese network in the SINT [22] architecture, which contains two identical convolutional

branches inherited from VGGNet [20]. This network aims to detect the most similar candidate to the target and functions as a generative subnet. Although training a CNN with multiple pooling operations makes the network more robust in dealing with object deformation and also helps to preclude over-fitting, pooling operation decreases the spatial resolution details. For instance, employing five pooling layers in the VGGNet would downsample the spatial information by a factor of 32. Therefore, the first two pooling layers in the VGGNet are intentionally retained and the other pooling layers, as well as all *fc* layers, are omitted. The architecture parameters of our baseline Siamese network is summarized in Table 1[1]. To benefit from multiple levels of abstract representation, three ROI (region-of-interest) pooling layers are applied to different layers. These ROI pooling layers transform features of multiple candidates with non-uniform sizes into the fixed-size feature maps simultaneously [33]. Due to the usage of the same input feature map for multiple candidates, ROI pooling layers speed up the performance of the network. Integrating these ROI pooling layers cannot be directly performed via a single concatenation operation, since their scales may differ considerably. Hence, a ℓ_2 normalization layer is employed after roipool1, roipool3, fc and concatenation layers to make the outputs of different layers limited to a unit sphere. Eventually, normalized concatenation layers are applied to a loss function, called margin contrastive loss. The underlying aim of this loss function is to make similar inputs embedded close together and to push dissimilar inputs apart [34]. This loss function is formulated as follows [35]:

$$\ell(u_i, u_j, t_{ij}) = \frac{1}{2} y D^2 + \frac{1}{2}(1 - c_{ij}) \max(0, \tau - D^2) \tag{3}$$

Where c_{ij} denotes the similarity of two inputs (i.e u_i and u_j) using a binary label; $D = \|F(u_i) - F(u_j)\|_2$ calculates the Euclidean distance between the two normalized outputs of the network; and τ indicates the minimum margin to separate two distinct inputs.

[1]Merely one branch of the network is reported.

Table 1: The architecture parameters of one branch of the Siamese network. As VGGNet, all convolutional layers are chased by a ReLU layer. Furthermore, roipool1, roipool3, fc and concat layers are also chased by a ℓ_2 normalization layer, where 'concat' denotes the concatenation layer. The kernel sizes are arranged as: (filter height, filter width, number of filters in the bank).

Layer	**Kernel size/Subdivisions**	**Stride/Spatial scale**	**Input**	**Output**	**Output size**
image	-	-	-	-	$512 \times 512 \times 3$
conv1 (2 sublayers)	$3 \times 3 \times 64$	1	image	conv1	$512 \times 512 \times 64$
maxpool1	2×2	2	conv1	maxpool1	$256 \times 256 \times 64$
conv2 (2 sublayers)	$3 \times 3 \times 128$	1	maxpool1	conv2	$256 \times 256 \times 128$
maxpool2	2×2	2	conv2	maxpool2	$128 \times 128 \times 128$
conv3 (3 sublayers)	$3 \times 3 \times 256$	1	maxpool2	conv3	$128 \times 128 \times 256$
conv4 (3 sublayers)	$3 \times 3 \times 512$	1	conv3	conv4	$128 \times 128 \times 512$
conv5 (3 sublayers)	$3 \times 3 \times 512$	1	conv4	conv5	$128 \times 128 \times 512$
roipool1[1]	7×7	0.25	conv4	roipool1	$1 \times 1 \times 25088$
roipool2	7×7	0.25	conv5	roipool2	$7 \times 7 \times 512$
roipool3[1]	7×7	0.25	conv5	roipool3	$1 \times 1 \times 25088$
fc	$7 \times 7 \times 4096$	1	roipool2	fc	$1 \times 1 \times 4096$
concat	-	-	(roipool1,roipool3,fc)	concat	$1 \times 1 \times 54272$

Once the Siamese network has been trained, the most likely candidates can be located using an inference scheme. However, because of evaluating candidates via a fixed Siamese network, it tends to drift in the cases of large appearance changes or coexisting of confusing objects. To tackle these challenges, we pursue two procedures: 1) Unlike previous Siamese-based trackers [7, 22] relying solely on the initial ground truth to find the target, in this work the best previous candidates are stored in a buffer and the new candidates are then assessed with the stored elements. To impede entering the occluded candidates into the buffer, buffer updating is restricted to a predefined condition; 2) Recently, much attention has been given to the use of sequence-specific information for fine-tuning a pretrained CNN [13, 12, 17, 18, 31, 36]. Integrating online fine-tuning strategy into a pretrained network can make the tracker adaptive to the large appearance changes of the target and thus enhance the tracker robustness. The trackers using this strategy locate the object of interest via the maximum positive score obtained from the fine-tuned network. Compared with the aforementioned strategy, in this work the WCNN is employed. The WCNN

[1]roipool1 and roipool3 are flattened.

functioning as a Discriminative subnet learns a weighting mechanism to discriminate positive candidates from negative ones by utilizing sequence-specific information. The weights are generated from the scores of the WCNN. This weighting mechanism can be utilized not only for down-weighting distracting candidates of the similar class but also for taking the target appearance variations into consideration. To achieve this goal, the WCNN constructed from two 1×1 convolutional layers is trained by a softmax logistic regression in the initial frame. During tracking, the WCNN would only be fine-tuned under peculiar conditions to preclude the tracker from drifting to similar objects. When the conditions are satisfied, the WCNN would be fine-tuned using features of positive and hard negative candidates ($F(u_{i,t})$). The WCNN in the weighting mechanism will be formulated and discussed in more details in section 3.3.

3.2.1. Training details

Training deep CNNs requires a large-scale dataset to achieve satisfactory results. ALOV [37] is one of the standard datasets which consists of 350 sequences with 65 different types of object. It also contains 13 different aspects of difficulty [37]. To unbiasedly train the Siamese network, all overlapped sequences between training and evaluation datasets should be omitted from the training dataset. Since we use the OTB100, OTB50 [23] and OTB2013 [24] datasets for evaluating our proposed tracker, 12 overlapped sequences should be eliminated. We cannot directly use standard datasets for training the proposed Siamese network, and hence we need to make some modifications to the standard datasets. Siamese networks require pairs of inputs images for training operation. To satisfy this requirement, the ground truth bounding box is extracted from a frame of a sequence for the first input. Then, multiple candidates are extracted from other random frames of that sequence for the second inputs. In summary, all candidates in the second inputs are compared with the ground truth in the first input. These candidates are supposed to be positive when their intersection-over-union (IoU) overlap ratios (with the ground truth box in the first input) exceed 0.7. On the other hand, they are considered to be negative

if their overlaps do not exceed 0.5.

The proposed Siamese network is initialized with the VGG-16 [20]. Fine-tuning the proposed Siamese network with the same learning rate for all layers may not lead to the optimal convergence [38]. Thus, layer-specific learning rate is utilized. Since initial layers of VGGNet extract low-level features that can be used either for image recognition or visual tracking in the same way, the first nine convolutional layers are frozen. The layer-specific learning rate for all other layers are set to 0.01 except for the fc layer, in which the learning rate is set to 1. All training details of the Siamese network are kept the same as in [22].

3.3. Inference and Updating Scenario

Inference. Once the Siamese network has been trained, it can be regarded as a matching function to locate the most similar objects to the predefined template. In this work, the most probable candidates are stored in the adaptable buffer B, which is used as a template in the matching function. This buffer is updated only when the final score ($\hat{u}_t^f$) reaches a predetermined threshold (see Algorithm 1). Thereby, occluded candidates are effectively filtered out. To carry out this strategy, the inference strategy in [39] is modified to take advantage of the adaptable buffer as follows:

$$\tilde{u}_{i,t} = (\eta M(u_g, u_{i,t}) + (\frac{1-\eta}{N}) \sum_{j=1}^{N} M(b_j, u_{i,t})) \tag{4}$$

where the matching function M is formulated as $M(p,q) = F(p)^T F(q)$; b_j indicates the j-th element of the buffer $B = \{b_1, ..., b_N\}$; η and u_g are a predetermined threshold and the ground truth in the initial frame; $u_{i,t}$ denotes i-th sampled candidate at time t. Once the sampled candidates have forwarded through the Siamese network, the WCNN can be fine-tuned using the output of the Siamese network while leveraging sequence-specific information. To be more specific, the aforementioned weighting mechanism takes the output of the Siamese network $F(u_{i,t})$ to fine-tune and train the WCNN. Ultimately, to down-weight background distracting candidates through the weighting mechanism, the

Algorithm 1 Online tracking algorithm

Input: Pretrained F-MEN, Pretrained Siamese network, The first ground truth bounding box u_g.

Output: Targat location $\hat{u}_t^f$,
Updated: Buffer B, A-MEN, WCNN

Feed u_g to the Siamese network and construct B using $F(u_g)$
Randomly initialize the weights of WCNN and A-MEN.
Generate candidates around the first target position u_g and pass them through the F-MEN and Siamese network.
Fine-tune WCNN using $\{F(u_{i,1})\}_{i=1}^{n_1^+}$ and $\{F(u_{i,1})\}_{i=1}^{n_1^-}$.
Fine-tune A-MEN using the outputs of the F-MEN.
$\mathcal{S}_{short} \leftarrow 1$, $\mathcal{S}_{long} \leftarrow 1$ and $\hat{u}_1^f \leftarrow u_g$.
for $t = 2, 3, ...$ **do**
    Apply search window centered at $\hat{u}_{t-1}^f$ to the MEN and output $\hat{u}_t^m$.
    Extract sample candidates around $\hat{u}_t^m$ and $\hat{u}_{t-1}^f$.
    Compute $\{F(u_{i,t})\}_{i=1}^{256}$ and also $\tilde{u}_{i,t}$ using Equation (4).
    Pass $\{F(u_{i,t})\}_{i=1}^{256}$ to WCNN and output $\{w_{i,t}\}_{i=1}^{256}$.
    Compute $\hat{u}_{i,t}$ and $\hat{u}_t^f$ using Equation(5).
    if $\hat{u}_t^f > 1.6$ or $t < 4$ **then**
        Update buffer B with the best $\{F(u_{i,t})\}$ corresponding to the $\hat{u}_t^f$.
        Generate candidates around the $\hat{u}_t^f$ and pass them through the F-MEN and Siamese network.
        $\mathcal{S}_{short} \leftarrow \mathcal{S}_{short} \cup \{t\}$, $\mathcal{S}_{long} \leftarrow \mathcal{S}_{long} \cup \{t\}$.
        if $|\mathcal{S}_{long}| > \tau_{long}$ **then** $\mathcal{S}_{long} \leftarrow \mathcal{S}_{long} \backslash \{\min_{\forall s \in \mathcal{S}_{long}} s\}$.
        if $|\mathcal{S}_{short}| > \tau_{short}$ **then** $\mathcal{S}_{short} \leftarrow \mathcal{S}_{short} \backslash \{\min_{\forall s \in \mathcal{S}_{short}} s\}$.
    if $\hat{u}_t^f < 1.6$ **then**
        Update WCNN and A-MEN using $n_t^+{}_{\forall s \in \mathcal{S}_{short}}$ and $n_t^-{}_{\forall s \in \mathcal{S}_{short}}$
    else if $mod(t, \tau_{int}) = 0$ **then**
        Update WCNN and A-MEN using $n_t^+{}_{\forall s \in \mathcal{S}_{long}}$ and $n_t^-{}_{\forall s \in \mathcal{S}_{short}}$

two scores are integrated as:

$$\hat{u}_{i,t} = \exp(\beta w_{i,t}) \odot \tilde{u}_{i,t} \tag{5}$$

where $\tilde{u}_{i,t}$ and $w_{i,t}$ indicate the i-th output scores of the Siamese and WCNN; $\hat{u}_{i,t}$ denotes the final state of i-th sampled candidate in frame t; β is a predetermined threshold. Finally, the final state of the target $\hat{u}_t^f$ is predicted by averaging the states of top five candidates ($\hat{u}_{i,t}$).

Updating. During tracking process, the object of interest undergoes a wide range of changes in illumination, pose, scale, orientation and shape. To overcome these challenges, the tracker should be updated over time. Hence, our tracker is equipped with two adaptable memories which are updated in short- and long-term strategies. These strategies can be dated back to the MDNet tracker [13]. In long-term strategy, updating is carried out every τ_{int} frames exploiting positive instances aggregated for τ_{long} time duration in the frame set $\mathcal{S}_{long}$. On the other hand, short-term updating is executed when $\hat{u}_t^f$ does not reach a predetermined threshold. The positive instances in short-term updating are aggregated for τ_{short} time duration in the frame set $\mathcal{S}_{short}$. Note that in both strategies we employ negative instances aggregated for τ_{short} time duration.

The aforementioned memories play a key role to update two different sections of our proposed tracker: (1) The A-MEN needs to be fine-tuned to properly model the motion of the target. Thereby, the first memory is constructed by the features of positive candidates in $\mathcal{S}_{long}$ and $\mathcal{S}_{short}$, which are calculated by the F-MEN; (2) To capture target appearance changes during tracking, the WCNN should be updated over time. To this end, $F(u_{i,t})$ is obtained for all positive and negative candidates in $\mathcal{S}_{long}$ and $\mathcal{S}_{short}$ to build the second memory.

4. Experiments

In this section, we first discuss experimental settings and implementation details. Then, we perform quantitative and qualitative experiments to assess our proposed tracker.

4.1. Experimental Settings

To demonstrate the robustness and adaptiveness of our proposed tracker against state-of-art trackers, we conduct extensive experiments on three challenging datasets including OTB100, OTB50 [23] and OTB2013 [24]. OTB50 and OTB2013 are subsets of OTB100 dataset. Note that OTB50 encompasses more challenging sequences than OTB2013. The videos in these datasets are labeled

with 11 attributes such as motion blur, occlusion, deformation, fast motion, background clutter and so forth. We adopt two metrics in OTB toolkit [23], i.e., distance precision (DP) and overlap success (OS), to draw precision and success plots and evaluate our tracker against the state-of-art competitors. The precision plot illustrates the ratio of frames where the distance of central pixels between the predicted and the ground truth bounding boxes does not reach to a predefined threshold. On the other hand, the success plot gauges the percentage of frames that their overlap criteria exceed a predefined threshold. By changing the threshold between 0 to 50 (0 to 1), the precision (success) plot is obtained. To rank trackers based on the distance precision and overlap success rates, the threshold of 20 pixels and *area under curve* (AUC) are utilized, respectively. We evaluate our proposed tracker against the state-of-art trackers including VITAL [40], MDnet [13], ADnet [41], HCFTs [42], SRDCF [43], CREST [5], SiamFC [7], ACFN [44], CFNet [6], Staple [45], LCT [46].

4.2. Implementation Details

To distinguish the positive candidates from the negative ones, the threshold R in Equation (2) is set to 12. The A-MEN is trained (fine-tuned) for 30 (10) epochs using stochastic gradient descent (SGD) with a global learning rate of 0.001, a momentum of 0.9, and a batch size of 8. The layer-specific learning rate for the first and second convolutional layers are set to 3 and 30, respectively. Overfitting issue is reduced using a weight decay of 0.0005. To preclude boundary discontinuities, the features calculated from the F-MEN are multiplied by a cosine window before inputting them into the A-MEN. The F-MEN remains fixed during the tracking process. WCNN follows the same training settings as A-MEN, except that the learning rate and momentum are set to 0.15 and 0.005, respectively. Moreover, each mini-batch in training the WCNN contains 32 positives and 96 hard negatives examples. In each frame, 250 candidates are sampled from the state variable x_t using Gaussian distribution around the expected locations centered at $\hat{u}_t^f$ and $\hat{u}_t^m$. The diagonal covariance matrix Σ is formulated as $\text{diag}(0.09v^2, 0.09v^2, 02.5)$, where v represents the mean of bound-

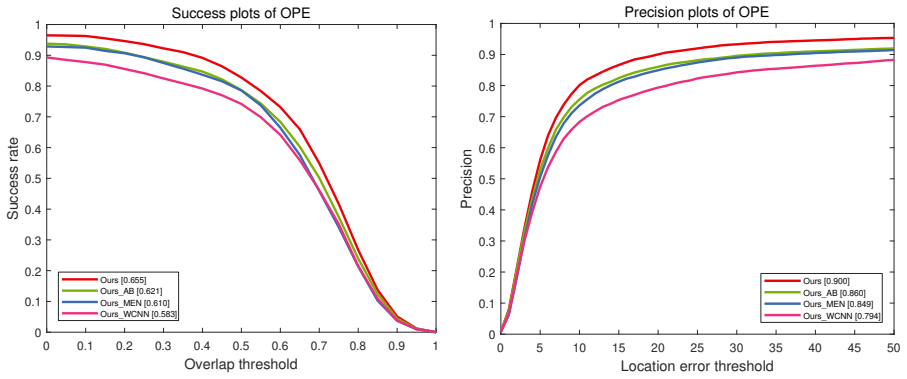
ing box size. To calculate the scale of new candidates in each frame, the ground truth scale is multiplied by a factor of 1.1 over time.

Candidate Generation. For training the WCNN, candidates are supposed to be positive when their IoU overlap ratios with the ground truth exceed 0.7. The numbers of positive and negative candidates (n_t^+ and n_t^-) for training the WCNN are set to 50 and 200. Likewise, they are considered to be negative if their IoU overlap ratios do not exceed 0.5. In fine-tuning phase, candidates are deemed to be negative if their IoU overlap ratios do not reach 0.3. Positive candidates follow the same assumption as before and the number of positive and negative candidates (n_1^+ and n_1^-) are set to 500 and 5000, respectively. Furthermore, to train and fine-tune the A-MEN, 50 and 5 positive candidates are accidentally chosen from n_t^+ and n_1^+ . The parameters used in long and short-term strategies, i.e., τ_{long}, τ_{short} and τ_{int} are set to 100, 20 and 10, respectively. τ in Equation (3) , N and η in Equation (4) and at last β in Equation (5) are set to 1, 35, 0.7 and 0.2, respectively.

4.3. Quantitative Comparison

Self-comparison. To investigate the contribution of MEN, WCNN and the adaptable buffer B, we evaluate the degraded versions of our tracker by removing them separately. In Figure 3, we denote these three versions as Ours-MEN, Ours-WCNN and Ours-AB. As depicted in Figure 3, each component in the proposed method plays an important role to improve the tracking accuracy. Particularly, the WCNN has the most influence on the tracking results.

Overall Results. We conduct *one pass evaluation* (OPE) of OTB toolkit to assess our tracker on large-scale OTB benchmark. Figure 4 illustrates the overall performance of the proposed tracker according to the DP and OS rates on OTB100, OTB50 and OTB2013. As shown in Figure 4, the proposed tracker performs satisfactorily in comparison with the others competitors. Figure 4 demonstrates that our tracker achieves better results in terms of OS criterion in comparison with the DP criterion. For the sake of clarify, in Table 2, the

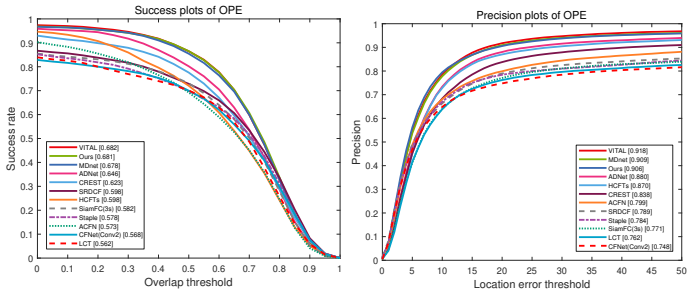


(a) Success Plot (b) Precision Plot

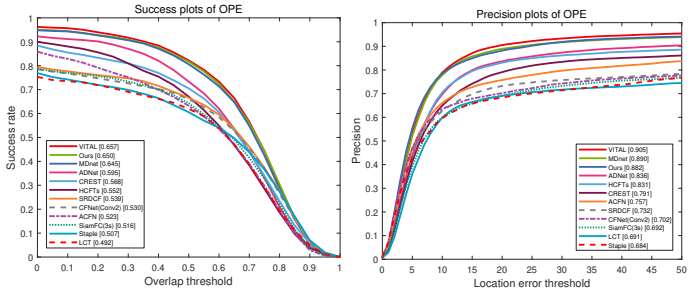
Figure 3: Self-comparison results in terms of success (a) and precision (b) plots on the subset of OTB100 dataset, including the 35 most challenging sequences.

quantitative results of precision and success plots are summarized. It is observed that we obtain a gain of %0.3 (%0.5, %0.4) AUC score over the MDNet on OTB100 (OTB50, OTB2013), respectively. This superiority is also maintained over the VITAL on OTB2013. In summary, our tracker gets the second best result in terms of overlap success rate and also the third best result in terms of distance precision rate. In terms of run-time speed, we compare our tracker with state-of-the-art trackers. Based on Table 2, our tracker runs at around 1.4 FPS, which is comparable with state-of-the-art non real-time trackers.

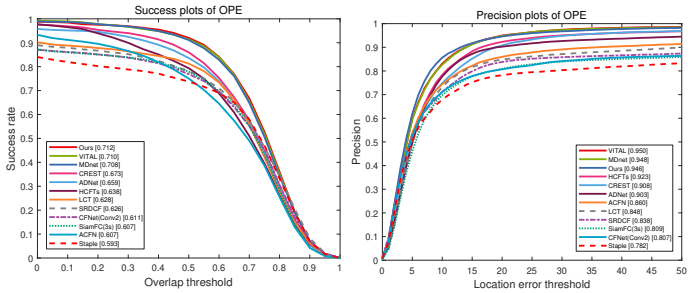
Performance Analysis per Attribute. In order to assess the proposed tracker ability in handling different challenges, the success plots for different subsets of the OTB dataset are shown in Figure 5. Each subset is categorized based on its attributes. Note that a sequence can take multiple tags in the process of tagging. Figure 5 illustrates the success plots of the attribute-based results on the OTB100 dataset. Figure 5 demonstrates that the proposed tracker achieves satisfactory results in tackling all reported challenges, especially in the cases of low resolution and occlusion challenges. For a clearer comparison, the AUC scores in Figure 5 and also the DP rates (at the threshold of 20 pixels on OTB100) are summarized in Figure 6. The proposed tracker ranks within top 3 on all 11 challenges. In terms of both success and precision plots, our tracker



(a) Results for OTB100



(b) Results for OTB50



(c) Results for OTB2013

Figure 4: Overall results of OPE according to the DP and OS rates on three datasets, including OTB100, OTB50 and OTB2013. The AUC scores and the DP rates (at the threshold of 20 pixels) are represented in the legend of subfigures. Only top twelve trackers are shown.

ranks the first in low resolution, in-plane rotation and occlusion challenges and obtains %1.1(%3.2), %0.6(%0.1) and %1(%0.2) improvements over the second best trackers, respectively. Due to the usage of hierarchical features, the highest improvements corresponds to the low resolution. Furthermore, Owing to the restrictions imposed on buffer updating, the proposed tracker can better handle

Table 2: Quantitative results on OTB100, OTB50 and OTB2013. Overlap success (OS) and distance precision (DP) are reported according to the AUC score and the error threshold of 20 pixels (in Figure 4), respectively. The run-time performance is based on GeForce GTX Titan X GPU except for our proposed method (GeForce GTX 1060 GPU) and VITAL method (Tesla K40c GPU). Top three results are indicated in red, blue and green font.

		OTB100		OTB50		OTB2013		
	Method	DP	OS	DP	OS	DP	OS	FPS
Non Real-time	Ours	90.6	68.1	88.2	65	94.6	71.2	1.4
	VITAL [13]	91.8	68.2	90.5	65.7	95.0	71.0	1.5
	MDNet [13]	90.9	67.8	89.0	64.5	94.8	70.8	1
	ADnet [41]	88	64.6	83.6	59.5	90.3	65.9	3
	CREST [5]	83.8	62.3	79.1	56.8	90.8	67.3	1
Real-time	SiamFC [7]	77.1	58.2	69.2	51.6	80.9	60.7	86
	CFNet [6]	74.8	56.8	70.2	53	80.7	61.1	75
	Staple [45]	78.4	57.8	68.4	50.7	78.2	59.3	80

occlusion than others. The proposed tracker also ranks the first on the success plots in the background clutter and illumination variation attributes and ranks the second in the out-of-plane rotation, scale variation and motion blur attributes.

4.4. Qualitative Comparison

Figure (7) illustrates the qualitative performance of the proposed tracker on some most challenging sequences from the OTB100 dataset in comparison with its main competitors, including VITAL [40], MDnet [13], ADnet [41], CREST [5], SiamFC [7], CFNet [6]. It is noteworthy that for challenging sequences such as *soccer*, *diving*, *skating2* and *motorRolling*, the trackers exploiting sequence-specific information (i.e., VITAL, MDnet and our proposed tracker) perform more robustly than the other trackers in dealing with occlusion, deformation, scale and illumination variations. However, VITAL and MDnet trackers tend to fail when the target moves fast, such as *biking* sequence. In contrast, our proposed tracker performs more accurately due to the use of MEN. In the box sequence, updating model with occluded objects makes MDnet tracker drift

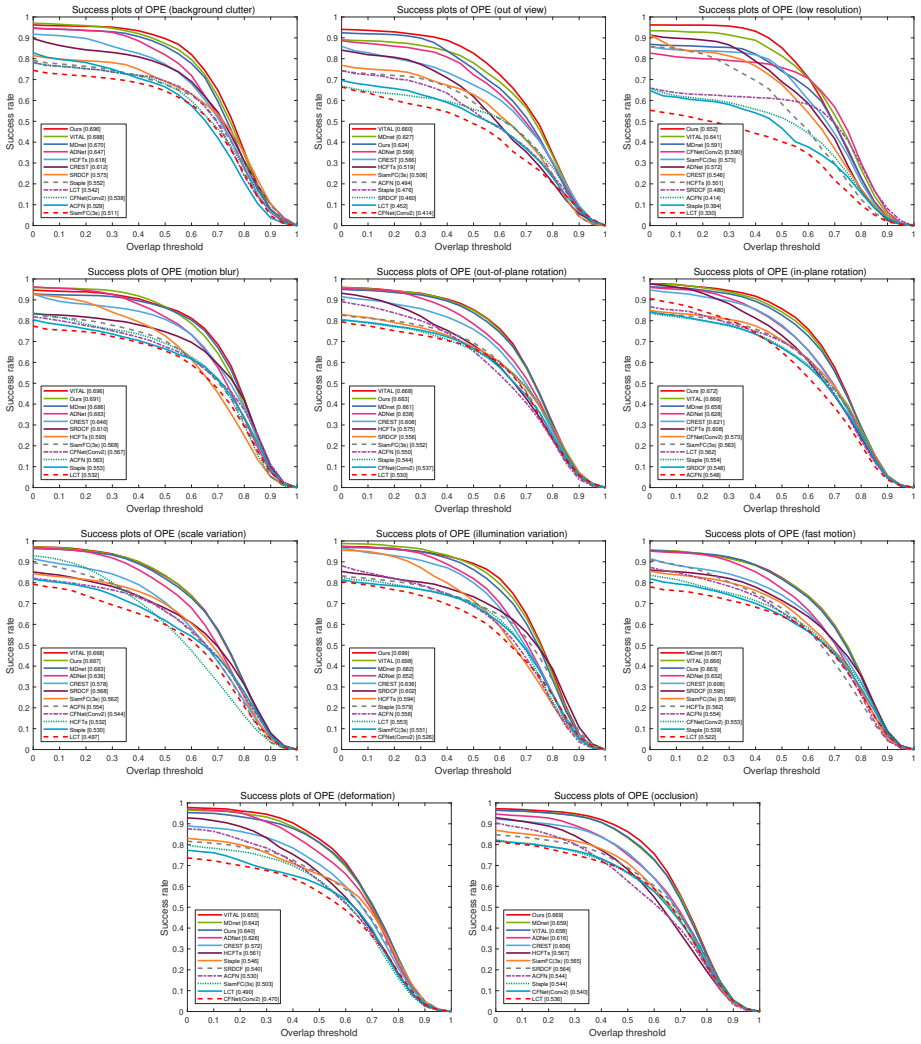
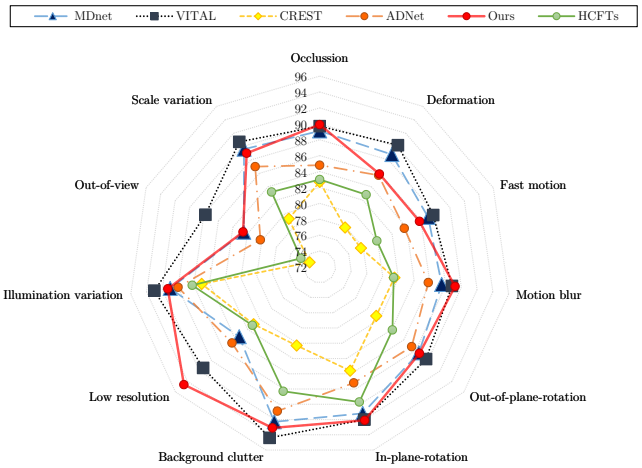
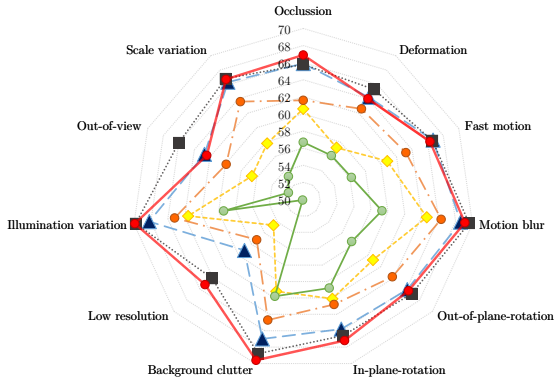


Figure 5: Success plots of OPE on 11 challenges, including fast motion, background clutter, motion blur, deformation, illumination variation, in-plane-rotation, low-resolution, occlusion, out-of-view, out-of-plane-rotation, scale variation.

away, whereas our tracker and VITAL perform favorably against long occlusion challenge. In the *matrix* sequence, where the background is quite cluttered, most approaches drift away. Again, the proposed tracker accurately localizes the target. In some rare cases, the object of interest quickly leaves field-of-view (fast motion and out-of-view occur simultaneously) and the tracker tends to be more susceptible to distracting similar objects. In addition, since the



(a) Attributes comparison in terms of success plots



(b) Attributes comparison in terms of precision plots

Figure 6: Distance precision rates (a) (at 20 pixels) and AUC score (b) of overlap success rates for the attribute-based results on the OTB100 dataset.

MEN only captures shallow features of target and outputs the most probable candidate, with target disappearance the MEN outputs the most similar object to the target. In such cases, the tracker considers these differences as appearance variations and drifts to distracting similar objects.

Figure 7: Qualitative performance of the proposed tracker, VITAL [40], MDnet [13], ADnet [41], CREST [5], SiamFC [7], CFNet [6] on some most challenging sequences from the OTB100 dataset (from top to bottom: *matrix*, *soccer*, *skating2*, *motorRolling*, *biking*, *diving*, *box*)

5. Conclusion

In this paper, we propose a robust tracking approach that addresses the motion and observation models simultaneously. In terms of the motion model, motion estimation network (MEN) is utilized to sample the most probable candidates. Then, to detect the best candidates among all sampled candidates, the Siamese network is trained offline. Due to the target appearance variations during the tracking process, each candidate is evaluated with an adaptable buffer containing the best selected previous candidates. In order to efficiently handle occlusion, buffer updating is limited to a predefined condition. Besides, to make the tracker more robust in dealing with coexisting of similar object and large appearance changes, a weighting CNN (WCNN) is exploited. This

WCNN employs sequence-specific information to down-weight distracting candidates. Experimental results verify that our approach performs satisfactorily against the state-of-art trackers.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[2] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587. doi:10.1109/CVPR.2014.81.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125. doi:10.1109/CVPR.2017.106.

[4] L. Leal-Taixé, C. Canton-Ferrer, K. Schindler, Learning by tracking: Siamese cnn for robust target association, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 33–40. doi:10.1109/CVPRW.2016.59.

[5] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, M.-H. Yang, Crest: Convolutional residual learning for visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2574–2583. doi:10.1109/ICCV.2017.279.

[6] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P. H. Torr, End-to-end representation learning for correlation filter based tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2805–2813. doi:10.1109/CVPR.2017.531.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. Torr, Fully-convolutional siamese networks for object tracking, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 850–865. doi:10.1007/978-3-319-48881-3_56.

[8] C. Ma, J.-B. Huang, X. Yang, M.-H. Yang, Hierarchical convolutional features for visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3074–3082. doi:10.1109/ICCV.2015.352.

[9] P. Pérez, C. Hue, J. Vermaak, M. Gangnet, Color-based probabilistic tracking, in: Proceedings of the European Conference on Computer Vision, 2002, pp. 661–675. doi:10.1007/3-540-47969-4_44.

[10] D. A. Ross, J. Lim, R.-S. Lin, M.-H. Yang, Incremental learning for robust visual tracking, International journal of computer vision 77 (2008) 125–141. doi:10.1007/s11263-007-0075-7.

[11] C. Sun, D. Wang, H. Lu, Occlusion-aware fragment-based tracking with spatial-temporal consistency, IEEE Transactions on Image Processing 25 (2016) 3814–3825. doi:10.1109/TIP.2016.2580463.

[12] L. Yang, R. Liu, D. Zhang, L. Zhang, Deep location-specific tracking, in: Proceedings of the 2017 ACM on Multimedia Conference, 2017, pp. 1309–1317. doi:10.1145/3123266.3123381.

[13] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4293–4302. doi:10.1109/CVPR.2016.465.

[14] J. Guo, T. Xu, Deep ensemble tracking, IEEE Signal Processing Letters 24 (2017) 1562–1566. doi:10.1109/LSP.2017.2749458.

[15] B. Han, J. Sim, H. Adam, Branchout: Regularization for online ensemble tracking with convolutional neural networks, in: Proceedings of IEEE

International Conference on Computer Vision and Pattern Recognition, 2017, pp. 2217–2224. doi:10.1109/CVPR.2017.63.

[16] K. Zhang, Q. Liu, Y. Wu, M.-H. Yang, Robust visual tracking via convolutional networks without training, IEEE Transactions on Image Processing 25 (2016) 1779–1792. doi:10.1109/TIP.2016.2531283.

[17] L. Wang, W. Ouyang, X. Wang, H. Lu, Visual tracking with fully convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3119–3127. doi:10.1109/ICCV.2015.357.

[18] S. Hong, T. You, S. Kwak, B. Han, Online tracking by learning discriminative saliency map with convolutional neural network, in: Proceedings of the International Conference on Machine Learning, 2015, pp. 597–606.

[19] P. Li, D. Wang, L. Wang, H. Lu, Deep visual tracking: Review and experimental comparison, Pattern Recognition 76 (2018) 323–338. doi:10.1016/j.patcog.2017.11.007.

[20] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of International Conference on Learning Representation, 2015.

[21] L. Wang, W. Ouyang, X. Wang, H. Lu, Stct: Sequentially training convolutional networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1373–1381. doi:10.1109/CVPR.2016.153.

[22] R. Tao, E. Gavves, A. W. Smeulders, Siamese instance search for tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1420–1429. doi:10.1109/CVPR.2016.158.

[23] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (2015) 1834–1848. doi:10.1109/TPAMI.2014.2388226.

[24] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2411–2418. doi:10.1109/CVPR.2013.312.

[25] E. Park, A. C. Berg, Meta-tracker: Fast and robust online adaptation for visual object trackers, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 587–604. doi:10.1007/978-3-030-01219-9_35.

[26] Y. Qi, S. Zhang, L. Qin, Q. Huang, H. Yao, J. Lim, M. Yang, Hedging deep features for visual tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence (2018) 1–1. doi:10.1109/TPAMI.2018.2828817.

[27] K. Chen, W. Tao, Once for all: A two-flow convolutional neural network for visual tracking, IEEE Transactions on Circuits and Systems for Video Technology 28 (2018) 3377–3386. doi:10.1109/TCSVT.2017.2757061.

[28] R. Pflugfelder, Siamese learning visual tracking: A survey, arXiv preprint arXiv:1707.00569 (2017).

[29] A. He, C. Luo, X. Tian, W. Zeng, A twofold siamese network for real-time object tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4834–4843. doi:10.1109/CVPR.2018.00508.

[30] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, H. Lu, Structured siamese network for real-time visual tracking, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 351–366. doi:10.1007/978-3-030-01240-3_22.

[31] H. Zhang, W. Ni, W. Yan, J. Wu, H. Bian, D. Xiang, Visual tracking using siamese convolutional neural network with region proposal and domain specific updating, Neurocomputing 275 (2018) 2645–2655. doi:10.1016/j.neucom.2017.11.050.

[32] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, in: Proceedings of the British Machine Vision Conference, 2014. doi:10.5244/C.28.6.

[33] R. Girshick, Fast r-cnn, in: Proceedings of IEEE International Conference on Computer Vision, 2015, pp. 1440–1448. doi:10.1109/ICCV.2015.169.

[34] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 1735–1742. doi:10.1109/CVPR.2006.100.

[35] S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 539–546. doi:10.1109/CVPR.2005.202.

[36] H. Fan, H. Ling, Sanet: Structure-aware network for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 2217–2224. doi:10.1109/CVPRW.2017.275.

[37] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah, Visual tracking: An experimental survey, IEEE transactions on pattern analysis and machine intelligence 36 (2014) 1442–1468. doi:10.1109/TPAMI.2013.230.

[38] B. Singh, S. De, Y. Zhang, T. Goldstein, G. Taylor, Layer-specific adaptive learning rates for deep networks, in: Proceedings of the IEEE International Conference on Machine Learning and Applications, 2015, pp. 364–368. doi:10.1109/ICMLA.2015.113.

[39] C. Jiang, J. Xiao, Y. Xie, T. Tillo, K. Huang, Siamese network ensemble for visual tracking, Neurocomputing 275 (2018) 2892–2903. doi:10.1016/j.neucom.2017.10.043.

[40] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. Lau, 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. doi:10.1109/CVPR.2018.00937.

[41] S. Y. J. C. Y. Yoo, K. Yun, J. Y. Choi, Action-decision networks for visual tracking with deep reinforcement learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1349–1358. doi:10.1109/CVPR.2017.148.

[42] C. Ma, J.-B. Huang, X. Yang, M.-H. Yang, Robust visual tracking via hierarchical convolutional features, IEEE Transactions on Pattern Analysis and Machine Intelligence (2018) 1–1. doi:10.1109/TPAMI.2018.2865311.

[43] M. Danelljan, G. Hager, F. Shahbaz Khan, M. Felsberg, Learning spatially regularized correlation filters for visual tracking, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4310–4318. doi:10.1109/ICCV.2015.490.

[44] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, J. Y. Choi, et al., Attentional correlation filter network for adaptive visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4828–4837. doi:10.1109/CVPR.2017.513.

[45] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, P. H. Torr, Staple: Complementary learners for real-time tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1401–1409. doi:10.1109/CVPR.2016.156.

[46] C. Ma, X. Yang, C. Zhang, M.-H. Yang, Long-term correlation tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5388–5396. doi:10.1109/CVPR.2015.7299177.