3-31-2021

# Learning to fuse asymmetric feature maps in Siamese trackers

Wencheng Han
*Beijing Institute of Technology*

Xingping Dong
*Inception Institute of Artificial Intelligence*

Fahad Shahbaz Khan
*Mohamed Bin Zayed University of Artificial Intelligence*

Ling Shao
*Inception Institute of Artificial Intelligence*

Jianbing Shen
*Beijing Institute of Technology*

# Learning to Fuse Asymmetric Feature Maps in Siamese Trackers

Wencheng Han*[1], Xingping Dong*[2], Fahad Shahbaz Khan[3], Ling Shao[2], Jianbing Shen†[2,1]
[1]Beijing Institute of Technology, [2]Inception Institute of Artificial Intelligence
[3]Mohamed Bin Zayed University of Artificial Intelligence,UAE
wencheng@bit.edu.cn, xingping.dong@gmail.com, fahad.khan@liu.se
ling.shao@ieee.org, shenjianbingcg@gmail.com

## Abstract

*Recently, Siamese-based trackers have achieved promising performance in visual tracking. Most recent Siamese-based trackers typically employ a depth-wise cross-correlation (DW-XCorr) to obtain multi-channel correlation information from the two feature maps (target and search region). However, DW-XCorr has several limitations within Siamese-based tracking: it can easily be fooled by distractors, has fewer activated channels and provides weak discrimination of object boundaries. Further, DW-XCorr is a handcrafted parameter-free module and cannot fully benefit from offline learning on large-scale data.*

*We propose a learnable module, called the asymmetric convolution (ACM), which learns to better capture the semantic correlation information in offline training on large-scale data. Different from DW-XCorr and its predecessor (XCorr), which regard a single feature map as the convolution kernel, our ACM decomposes the convolution operation on a concatenated feature map into two mathematically equivalent operations, thereby avoiding the need for the feature maps to be of the same size (width and height) during concatenation. Our ACM can incorporate useful prior information, such as bounding-box size, with standard visual features. Furthermore, ACM can easily be integrated into existing Siamese trackers based on DW-XCorr or XCorr. To demonstrate its generalization ability, we integrate ACM into three representative trackers: SiamFC, SiamRPN++ and SiamBAN. Our experiments reveal the benefits of the proposed ACM, which outperforms existing methods on six tracking benchmarks. On the LaSOT test set, our ACM-based tracker obtains a significant improvement of 5.8% in terms of success (AUC), over the baseline.*

## 1. Introduction

Visual tracking is a challenging problem, where the task is to estimate the state of an arbitrary target in each frame of a video, given only its location in the initial frame. Recently, trackers based on Siamese networks have gained attention due to their combined advantage of high speed and tracking performance. The pioneering method, SiamFC [2], utilizes Siamese networks to extract deep convolutional features from the template in the initial frame of a video and instances inside the search regions of other frames. A cross correlation layer (XCorr) is then used to compute the similarity between the template and instances. Consequently, the instance with the highest similarity score is considered the target. The XCorr in SiamFC produces a single-channel response map and assumes the target is located near the highest response. As an extension, SiamRPN [26] formulates the tracking problem as one-shot detection. It introduces a region proposal network (RPN) [34] and utilizes up-channel cross correlation (UP-XCorr). However, UP-XCorr imbalances the parameter distribution, making the training optimization hard. To address this issue, SiamRPN++ introduces a depth-wise correlation (DW-XCorr) to efficiently generate a multi-channel correlation feature map. Due to its efficiency, several recent Siamese trackers [5, 11, 13, 47, 50] also employ DW-XCorr in their frameworks.

As discussed above, most recent Siamese trackers employ DW-XCorr to compute the similarity between the template and instances. However, both DW-XCorr and its predecessor XCorr are handcrafted parameter-free modules and are not able to fully benefit from large-scale offline learning. DW-XCorr has several limitations in the context of tracking. First, it produces similar correlation responses for the target and distractors of homogeneous appearance. To demonstrate this, we analyze the similarity between DW-XCorr features of a target and its distractors in Fig. 1a. The heatmap is generated by performing an L1 normalization ($\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$, where $x$ is a pixel in the correlation feature map and $n$ is the number of channels) on every pixel in the DW-XCorr features. As can be seen, DW-XCorr produces high responses (*i.e.* feature norms) not only near the

. *Equal contribution. † Corresponding author.

. Our codes are available at: https://github.com/wencheng256/SiamBAN-ACM

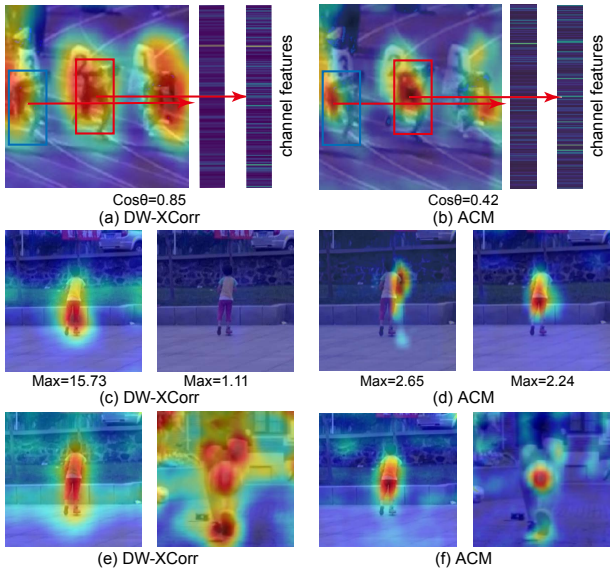


Figure 1. **Comparison between DW-XCorr and ACM in terms of being fooled by distractors (first row), information distribution across channels (second row) and background suppression to better discriminative target boundaries (third row).** DW-XCorr produces similar responses for distractors and the target (Fig. 1**a**). In contrast, ACM produces more distinct responses (Fig. 1**b**). In both cases (a and b), red arrows indicate the feature vectors extracted from the correlation feature maps of the corresponding pixels, followed by computing the cosine similarity ($cos\theta = \frac{A \cdot B}{\|A\| \|B\|}$) between the two feature vectors ($A$ and $B$). Only a few channels of DW-XCorr have high response when tracking a desired target (Fig. 1**c**). Instead, more channels of ACM map carries high response with different semantic information, such as top right corner (left) or center of target (right), as shown in Fig. 1**d**. We show two example feature channels for DW-XCorr and ACM. DW-XCorr maps are blurry and do not accurately capture shape of target (Fig. 1**e**). In comparison, AC maps suppresses the background, providing clear boundaries of the target (Fig. 1**f**).

target (the red rectangle), but also near other instances. We compute the cosine similarity between the target and one distractor (the green rectangle) and observe a high value ($cos\theta > 0.8$), indicating that DW-XCorr produces similar results for both. This makes it difficult for RPN to effectively discriminate the desired target from distractors.

The second limitation is that only a few channels in the DW-XCorr feature map are activated, *i.e.* have a high response when tracking a particular target [25]. To perform cross-correlation, features of different targets are desired to be orthogonal and distributed in different channels, so that correlation feature channels of different targets are suppressed and only a few channels of the same target are activated. The suppressed channels are unable to help RPN in making robust and precise predictions and can reduce the capacity of the model. As shown in Fig. 1c, the maximum value of a channel with middle response is significantly lower than the global maximum value. This indicates that these channels contribute little to the final predictions. Last, DW-XCorr often produces responses at irrelevant background. As a consequence, correlation maps are often blurry and do not have clear boundaries, as shown in Fig. 1e. This is likely to hinder RPN from making accurate and robust predictions.

The aforementioned shortcomings of DW-XCorr and its predecessor XCorr, within Siamese-based trackers, motivate us to look into designing a new module that learns to fuse feature maps by benefiting from offline learning on large-scale data. In case of two feature maps (*e.g.* the template and sub-window in a search image) having the same size, a straightforward way is to concatenate (fuse) them and then learn a method for joint training by adding convolutional layers. Here, the additional convolutional layers can learn to discriminate the target and background. However, such a concatenation strategy is non-trivial in the case of Siamese-based trackers since the two feature maps are of different sizes (height and width). Further, the concatenation of feature maps of different sizes is desired to be performed in an efficient manner to meet the real-time requirements during inference.

### 1.1. Contributions

We introduce a novel module, called the *asymmetric convolution (ACM)*, that avoids the need for the feature maps to be of the same size during concatenation. Our ACM decomposes the convolution operation on a concatenated feature map into two mathematically equivalent operations. First, it performs convolutions on two feature maps independently using kernels of the same size as that of the template feature map. Then, it performs a summation on the resulting feature maps, through broadcasting [15]. By utilizing the broadcasting of matrix addition, we efficiently compute the summation on these different-sized feature maps.

The proposed ACM produces more discriminative features, as shown in Fig. 1b, with respect to the target (the red rectangle) and distractors (the green rectangle). This enables the tracker to make more robust predictions. Further, the maximum values of different channels in our ACM are closer, which indicates that more channels carry useful information, as shown in Fig. 1d. At the same time, ACM can effectively suppress background, thereby providing clear boundaries for the target, as in Fig. 1f. We validate these advantages by conducting an extensive analysis on 50k different image pairs from the LaSOT train set [12]. Details are presented in §3.2.

In addition to overcoming the aforementioned limitations of DW-XCorr, the proposed ACM is flexible and can also incorporate useful additional information. Here, we incorporate prior information in the form of bounding-box (b-box) size (height and width) from the initial frame in a

video. This prior information helps to overcome the lack of accurate target-box locations in the template image, thereby providing guidance to the RPN heads. Furthermore, we show the generalization ability by replacing the standard DW-XCorr or XCorr with our ACM in three representative Siamese-based trackers: SiamFC [2], SiamRPN++ [25] and SiamBAN [5]. Comprehensive experiments on six tracking benchmarks show the benefits of our ACM, leading to favorable performance against existing methods. On the large-scale LaSOT test set [12], our ACM-based trackers (SiamFC-ACM, SiamRPN++ACM and SiamBAN-ACM) achieve relative gains of 8.6%, 5.7% and 11.3%, in terms of area-under-the-curve (AUC), over their respective baselines (SiamFC, SiamRPN++ and SiamBAN).

## 2. Related Work

Recently, deep learning has pervaded computer vision with great success in a variety of tasks, including object tracking [1, 2, 6, 18, 21, 31–33, 37, 40, 42, 51]. Several deep learning-based trackers learn a classifier online to distinguish the target from the background and distractors [6, 31, 36, 39, 42]. The MDNet [31] tracker employs a CNN trained offline from multiple annotated videos. During evaluation, it learns a domain-specific detector online to discriminate between the background and foreground. ATOM [6] comprises two dedicated components: target estimation, which is trained offline, and classification trained online. DiMP [3] employs a meta-learning based architecture, trained offline, that predicts the weights of the target model. The recently introduced KYS [4] extends DiMP by exploiting scene information to improve the results.

Several existing deep trackers [2, 5, 13, 25, 26, 47] are based on Siamese networks and focus on learning a universal discriminator during large-scale offline learning. These trackers formulate the task as a general similarity computation between the target template and the search region. The pioneering work, SiamFC [2], introduced the XCorr layer to combine feature maps and can run at a speed of 100 frames per second (FPS). Since this work, several researchers have tried to further mine the potentiality of the Siamese framework by designing different Siamese architectures [10, 16, 43, 52], using a powerful training loss [7], learning efficient Siamese networks [28], learning a dynamic network [14], utilizing deep reinforcement learning [8, 9, 20], and so on [27, 35, 44, 48]. SiamRPN++ [25] and SiamDW [53] overcome the issues of previous Siamese-based trackers that restrict them to using only relatively shallow networks. Specifically, they address the problems stemming from destroying the strict translation invariance and introduce modern deep networks, such as, ResNet [17], and ResNeXt [46], into Siamese trackers. SiamRPN++ utilizes a depth-wise correlation (DW-XCorr) to efficiently generate a multi-channel correlation feature map. The recently introduced SiamBAN [5] and SiamCAR [13] also employ DW-XCorr and use an anchor-free strategy to predict bounding-boxes (b-boxes) directly without pre-defined anchor boxes.

**Our Approach:** As discussed earlier, most recent Siamese trackers typically employ a handcrafted module, DW-XCorr, to compute the similarity between the template and instances. Both DW-XCorr and its predecessor XCorr are not able to fully benefit from large-scale offline learning and have several limitations, including being easily fooled by distractors and providing weak discrimination of the object boundaries. To address these issues, we propose a new module (ACM) that learns to better capture semantic information from large-scale data during offline training. Our ACM produces more discriminative features with respect to the target and distractors, contains more activated channels carrying useful information and effectively suppresses the background, thereby providing clear boundaries of the target. Furthermore, our ACM is flexible and generic, enabling easy integration into existing Siamese trackers. We show this by integrating our ACM into three Siamese trackers and demonstrate its effectiveness on six benchmarks.

## 3. Method

### 3.1. Siamese Networks for Tracking

Siamese networks formulate the tracking task as learning a general similarity map between the feature maps extracted from the target template and the search region. When certain sliding windows in the search region are similar to the template, responses in these windows are high [2]. These networks are designed as Y-shaped, with two branches: one for the template $\mathbf{z}$ and the other for the search region $\mathbf{x}$. The two branches share the same network $\varphi$ with parameters $\theta$ and produce two feature maps $\bar{\mathbf{z}} = \varphi(\mathbf{z};\theta) \in \mathbb{R}^{C\times\eta\times\omega}$ and $\bar{\mathbf{x}} = \varphi(\mathbf{x};\theta) \in \mathbb{R}^{C\times H\times W}$. These two feature maps have the same channel number ($C$) but different sizes ($\eta \times \omega$ *vs.* $H \times W$), where $\eta \leq H$ and $\omega \leq W$. Then, a function $f$ is used to combine the feature maps and generate a similarity map $\mathbf{c} \in \mathbb{R}^{1\times(H-\eta+1)\times(W-\omega+1)}$, where the center of the target is most likely found at the position with the highest response. Usually, $f$ is an XCorr operation $*$ between $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$. The formulation is as follows:

$$\mathbf{c} = f(\bar{\mathbf{z}}, \bar{\mathbf{x}}) = \varphi(\mathbf{z};\theta) * \varphi(\mathbf{x};\theta). \tag{1}$$

To further improve the performance of Siamese-based trackers, SiamRPN [26] adds region proposal network (RPN) [34] to generate bounding-boxes (b-boxes) for each frame of a tracking sequence.The RPN contains two XCorr modules to extract correlation maps and two heads on them to perform anchor classification and regression, respectively. This is different to previous Siamese trackers, such as SiamFC, where the b-box is not explicitly regressed and is typically set based on the size that best matches the search

region. While SiamRPN utilizes an RPN, it employs up-channel cross correlation (UP-XCorr), which imbalances the parameter distribution, making the training optimization difficult. SiamRPN++ [25] addresses this issue by introducing a depth-wise correlation (DW-XCorr) to efficiently generate a multi-channel correlation feature map $\mathbf{c}_{dw}$, as in Fig. 2a. The formulation is as follows:

$$\begin{aligned} \mathbf{c}_{dw} &= f(\bar{\mathbf{z}}, \bar{\mathbf{x}}) = \bar{\mathbf{z}} \otimes \bar{\mathbf{x}}; \\ \mathbf{c}_{dw} &\in \mathbb{R}^{N \times (H-\eta+1) \times (W-\omega+1)}, \end{aligned} \tag{2}$$

where $\otimes$ is a depth-wise convolution [19] of two feature maps, and $N$ is the number of channels. Then, the features are fed into the RPN heads to produce the final tracking b-box. The RPN heads are usually constructed with sequences of $1 \times 1$ convolutional layers, including the classification module $\mathcal{H}_{cls}$, which predicts the classification score of each b-box candidate, and the regression module $\mathcal{H}_{loc}$, which obtains the details (size in terms of width and height) of each b-box. By applying these heads to the correlation maps, we can obtain the score map $\mathbf{m}_{cls} \in \mathbb{R}^{2A \times (H-\eta+1) \times (W-\omega+1)}$ and b-box map $\mathbf{m}_{loc} \in \mathbb{R}^{4A \times (H-\eta+1) \times (W-\omega+1)}$:

$$\mathbf{m}_{cls} = \mathcal{H}_{cls}(\mathbf{c}_{dw}^{cls}; \theta_{cls}), \quad \mathbf{m}_{loc} = \mathcal{H}_{cls}(\mathbf{c}_{dw}^{loc}; \theta_{loc}). \tag{3}$$

As we can see, the fusion method $f$ is crucial for Siamese-based trackers. However, both the XCorr and depth-wise XCorr (DW-XCorr) are parameter-free methods and therefore cannot fully benefit from large-scale training. Further, they have several limitations as described in §1. Our asymmetric convolution module (ACM) addresses these limitations by introducing an asymmetric convolution (AC) as $f(\bar{\mathbf{z}}, \bar{\mathbf{x}}; \theta_{ac})$. With the parameter $\theta_{ac}$, AC can be optimized during training and finds the a better way to fuse $\bar{\mathbf{z}}$ and $\bar{\mathbf{x}}$.

### 3.2. Asymmetric Convolution

Different from handcrafted methods (*e.g.*, DW-XCorr and XCorr) for fusing features in Siamese networks, we look into how to concatenate two different-sized feature maps and learn a fusion during offline training on large-scale data. Learning to fuse feature maps during offline training is expected to provide rich prior information, enabling the fusion method to better adapt to different challenging situations, such as motion blur, deformation, fast motion and clutter. However, an efficient direct concatenation of these feature maps is challenging due to the different sizes of the template and search image. To this end, we investigate the problem of efficiently fusing feature maps of different sizes. A straightforward way (Fig. 2b) is to first split the the search region feature map into $n$ sub-windows of the same size as that of the template feature map. It is woth noting that every sub-window is a sliding window here. Then, $n$ different sub-windows and the template can

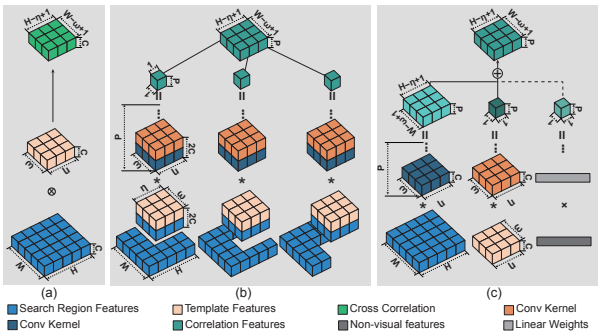


Figure 2. **Comparison of (c) AC with (a) DW-XCorr and (b) a naive strategy to fuse different-sized feature maps. (a):** DW-XCorr uses a $C$ channel feature map extracted from the template as kernel and convolves instance feature maps in a depth-wise manner to generate a $C$ channel correlation feature map. **(b):** A naive strategy to perform concatenation on different-sized feature maps (template and search region) is to first split the search region feature map into $n$ sub-windows of the same size as that of the template feature map. Then, $n$ different sub-windows and the template are concatenated along channel axis, followed by a convolution to generate a new feature during offline training. **(c):** AC efficiently concatenates different-sized feature maps by first separately convolving the two feature maps (template and search) using kernels of same size as that of the template feature map. Then, it computes summation on these different-sized feature maps through broadcasting. In addition, our AC possesses the ability to incorporate useful non-visual features (dashed line), such as b-box size.

be concatenated along the channel axis, followed by a convolution operation to produce a new feature $\mathbf{v}_i$. However, such a strategy (Fig. 2b) based on direct convolution on the concatenated feature map is computationally expensive, since the convolution operation is required to be repeated for each sub-window. To circumvent this problem, we introduce a mathematically equivalent procedure, called the asymmetric convolution (AC), that replaces this direct convolution on the concatenated feature map with two independent convolutions (Fig. 2c). For a sub-window $n$, our AC, comprising two independent convolutions followed by a summation, is mathematically equivalent to the direct convolution on the concatenated feature map:

$$\begin{aligned} \mathbf{v}_i &= \begin{bmatrix} \theta_z & \theta_x \end{bmatrix} * \begin{bmatrix} \bar{\mathbf{z}} \\ \bar{\mathbf{x}}_i \end{bmatrix} = \theta_z * \bar{\mathbf{z}} + \theta_x * \bar{\mathbf{x}}_i; \\ \bar{\mathbf{x}}_i &\in \mathbb{R}^{C \times \eta \times \omega}, \theta_z, \theta_x \in \mathbb{R}^{P \times C \times \eta \times \omega}, \mathbf{v}_i \in \mathbb{R}^{P \times 1 \times 1}, \end{aligned} \tag{4}$$

where $\bar{\mathbf{x}}_i$ is a window of $\bar{\mathbf{x}}$, $\theta_z$ is the kernel applied to $\bar{\mathbf{z}}$, and $\theta_x$ is that applied to $\bar{\mathbf{x}}$. After the convolution operation, the result $\mathbf{v}_i$ has a shape of $P \times 1 \times 1$. The left side of Eq. 4 is a convolution on a concatenated feature map of $\bar{\mathbf{z}}$ and $\bar{\mathbf{x}}_i$, and it is equivalent to the right side, *i.e.*, two independent convolutions and a summation. Next, we collect the features of all windows inside $\bar{\mathbf{x}}$ to formulate a new feature map $\mathbf{v}$, as

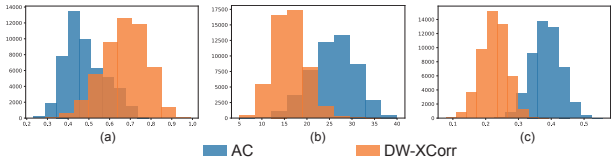


Figure 3. **Comparison between DW-XCorr and AC in terms of (a and b) producing more discriminative features for targets and distractors to avoid being fooled by the distractors and (c) information distribution across correlation channels.** Comparison is performed on 50K different image pairs from LaSOT train set. **(a):** Cosine similarity between targets and distractors based on DW-XCorr and AC feature maps, respectively. **(b):** Same as (a), except cosine similarity is replaced by Euclidean distance. In (b), the correlation feature maps are first normalized between [0,1] and then the Euclidean distance is computed between targets and distractors. **(c):** Average values over all maximum feature values of channels for DW-XCorr and AC, respectively. In each case, the maximum feature values are obtained by first performing a normalization (dividing the values by their global maximum value).

follows:

$$\begin{aligned}\mathbf{v} &= \{\mathbf{v}_i \mid i \in [1,n]\} \\ &= \{\theta_z * \bar{\mathbf{z}} + \theta_x * \bar{\mathbf{x}}_\mathbf{i} \mid i \in [1,n]\} \\ &= \theta_z * \bar{\mathbf{z}} +_b \theta_x * \bar{\mathbf{x}},\end{aligned} \tag{5}$$

where $+_b$ is a summation with broadcasting. We utilize the broadcasting method since it efficiently conducts arithmetic operations on matrices with different shapes and is widely available in scientific computing packages, including Numpy [15] and Pytorch [38]. Through broadcasting, engines allow the dimensions of arrays to differ. Specifically, arrays with smaller sizes are virtually duplicated (that is, without copying any data in the memory and thus introducing little computational burden), so that the shapes of the operands match [15]. Moreover, all sub-windows inside $\bar{\mathbf{x}}$ share the same convolution. Therefore, we replace $\{\theta_x * \bar{\mathbf{x}}_i | i \in [1,n]\}$ with $\theta_x * \bar{\mathbf{x}}$ for simplicity. In this way, we perform a convolution operation on two feature maps with different shapes, simultaneously. After applying a ReLU activation function, we obtain a new fusion $f(\bar{\mathbf{z}}, \bar{\mathbf{x}}; \theta_{ac})$ which can be optimized during training:

$$\begin{aligned}\mathbf{c}_{ac} &= f(\bar{\mathbf{z}}, \bar{\mathbf{x}}; \theta_{ac}) \\ &= ReLU(\theta_z * \bar{\mathbf{z}} +_b \theta_x * \bar{\mathbf{x}}); \\ \mathbf{c}_{ac} &\in \mathbb{R}^{C\times(H-\eta+1)\times(W-\omega+1)}.\end{aligned} \tag{6}$$

As discussed earlier, our AC benefits from the offline training and alleviates the limitations of DW-XCorr. To demonstrate that AC produces more discriminative features for the targets and distractors than XCorr, we perform an experiment in which we compute the cosine similarity between targets and distractors based on the AC and XCorr feature maps, respectively on 50k different image pairs from the LaSOT dataset. We set the target to be at the center of the search region and select the features located at the center of the AC and DW-XCorr maps to represent it. Then, we find the maximum response outside the b-box region and select features at this point to represent the distractor. Finally, the cosine similarity between the target and distractor features is computed to evaluate the discriminative ability of AC and DW-XCorr. Fig. 3a shows that AC maps are less affected by distractors, producing more discriminative features, compared to DW-XCorr. Fig. 3b shows a similar comparison but from a different perspective, where cosine similarity is replaced with the euclidean distance. Here, the correlation feature maps are first normalized between [0,1] and then the Euclidean distance is computed between targets and distractors. Further, AC maps contain more semantic information than DW-XCorr, as shown earlier in Fig. 1b. We also validate, on same 50k image pairs from LaSOT, that AC channels provide more diversity when extracting correlation information, compared to DW-XCorr. We first normalize AC and DW-XCorr by dividing them by their global maximum value, and then calculate maximum values of each channel. Finally, average values over all channels are used to draw a comparison, shown in Fig. 3c. Lastly, AC maps suppresses influence of irrelevant background better, compared to DW-XCorr, as shown earlier in Fig. 1f. This helps RPN heads to more accurately predict the b-boxes.

### 3.3. Incorporating Prior Non-Visual Information

As discussed earlier, our ACM is flexible and can incorporate additional (non-visual) information. Here, we show the integration of prior information in the form of target b-box size (width and height) from the initial frame. It is worth noting that traditional RPN head has no exact prior information about the target b-box which can be of arbitrary shape. ACM can provide such additional prior information, in terms of a b-box size, to the RPN head for accurate target localization. However, a b-box size is a one-dimensional feature and cannot be fed directly into 2D convolutional networks. Here, we regard a b-box size as a specific image feature with a size of $C_b \times 1 \times 1$, where $C_b$ is the channel number. In this way, we utilize ACM to fuse useful prior information, such as b-box size, with standard high-dimensional visual features representing template and search regions.

We use the b-box size information from the initial frame in our tracking framework to distinguish features belonging to the target and provide guidance to the RPN heads:

$$\begin{aligned}\mathbf{c}_{ac} &= f(\bar{\mathbf{z}}, \bar{\mathbf{x}}, \mathbf{B}; \theta_{ac}, \theta_{box}) \\ &= ReLU(\theta_z * \bar{\mathbf{z}} +_b \theta_x * \bar{\mathbf{x}} +_b \eta(\mathbf{B}, \theta_{box}));\end{aligned} \tag{7}$$

Here, $B$ is the b-box of the initial frame and $\eta$ is a three-layer fully-connected network with parameters $\theta_{box}$. Since the target in the template is always at the center of the image, we only use the width and height of the b-box.

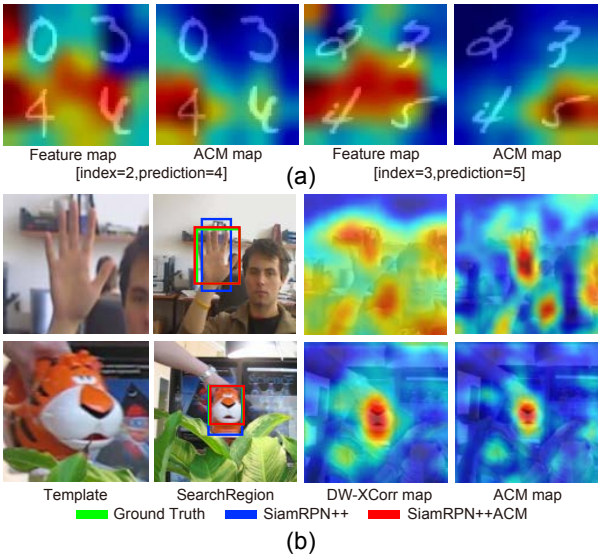


Figure 4. **(a): Effectiveness of our ACM** in fusing additional information (single number to indicate digit location) with visual feature maps for the task of digit prediction on MNIST. Here, "index" means the position to predict, and "prediction" is the predicted digit at this position. Indexes are 0,1 in the first row and 2,3 in the second row of the $2 \times 2$ matrix. The colors, superimposed on the images, are responses of feature maps where high responses are represented by warm colors. **(b): Tracking comparison between our ACM-based tracker (SiamRPN++ACM) and the baseline (SiamRPN++)** on example frames, where the target is only part of an object (*e.g.*, part of hand or body). Here, we also show DW-XCorr and ACM feature maps of the baseline and SiamRPN++ACM, respectively. Each feature map shown is obtained by taking the L1 norm of each pixel in the respective feature map. Our ACM map is able to focus on regions belonging to the target. Moreover, the integration of non-visual 1D b-box size features provides useful prior information to the RPN heads, leading to more accurate predictions.

Fig. 4(b) shows a tracking comparison between our ACM-based tracker and the baseline (using DW-XCorr) on example frames, where target is only part of an object (*e.g.*, part of hand or body).

To further demonstrate the effectiveness of our fusion, we conduct a simple experiment for digit prediction on MNIST dataset [24]. First, we concatenate number images from MNIST into a $2 \times 2$ matrix and randomly generate an index of 0-3 to indicate the position of the numbers. Then, we design a network similar to AlexNet to predict the number at a given position. To incorporate the index information (a single number), we extract the index features using a three-layer fully-connected network and fuse them with the feature map of a matrix image using our ACM. We then feed the fused features into a prediction network. As shown in Fig. 4(a), high responses are uniform without using index information. After integrating index information using

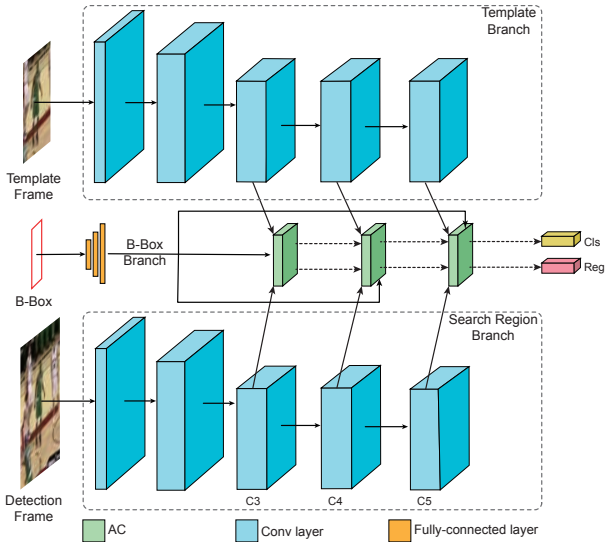


Figure 5. **Overview of our tracker (SiamBAN-ACM)** which integrates ACM, in place of DW-XCorr, in the baseline SiamBAN.

ACM, they are more concentrated around the target positions. Even though we only give the network a single index number, it is able to better discriminate target position with emphasis to the region belonging to the target. As a result, our network correctly predicts the number at given position.

### 3.4. ACM for Visual Tracking

The proposed ACM is generic and can be easily integrated into existing Siamese trackers. Here, we integrate ACM into three trackers: SiamFC [2], the recently introduced SiamRPN++ [25] and SiamBAN [5]. For SiamFC, we replace its XCorr with ACM, whereas for SiamRPN++ and SiamBAN we replace their DW-XCorr with ACM. The resulting ACM-based trackers are named, SiamFC-ACM, SiamRPN++ACM and SiamBAN-ACM, respectively.

**Our SiamFC-ACM.** The original SiamFC [2] employs XCorr to produce a single-channel response map. We use the same network as the original SiamFC to extract features, and feed the feature maps produced by the template and search region branches into ACM, producing a correlation map with a single channel. The position with the highest response is then set as the predicted target center.

**Our SiamRPN++ACM.** The original SiamRPN++ [25] is the first to introduce DW-XCorr into Siamese trackers. For SiamRPN++ACM, we replace the DW-XCorr in the original SiamRPN++ with our ACM. Specifically, ACM fuses the features from the three branches (template, search region and b-box) to generate a correlation feature map, as shown in Fig. 5. The b-box branch uses three fully-connected layers to generate a target location feature map ($1 \times 1 \times 256$). Then, we apply two $5 \times 5$ convolutions without padding to the template and search region feature maps to obtain semantic feature maps. Consequently, the

summation of the three feature maps (*i.e.* template, search region and b-box maps) is then batch normalized and used as input to the RPN heads. The template and initial b-box are fixed during inference and the three branches remain independent until the broadcasting summation. Thus, we can cache the two branches (template and b-box) to save computational cost. In this way, the additional computational cost introduced by ACM is only a single convolution on the search region, thereby causing no significant degradation to the overall inference speed.

**Our SiamBAN-ACM.** The recent SiamBAN [5] does not employ pre-defined anchors, enabling it to perform better and faster than its baseline SiamRPN++. To obtain SiamBAN-ACM, we apply same changes (replacing DW-XCorr with ACM) to the baseline SiamBAN as described above for SiamRPN++ACM.

## 4. Experiments

We perform comprehensive experiments on six tracking benchmarks: OTB-100 [45], UAV123 [29], TrackingNet [30], VOT2016, VOT2019 [23] and LaSOT [12]. A well-documented and complete training and inference code will be publicly released.

**Implementation Details.** Our ACM-based tracking frameworks are implemented using the Pytorch tracking platform PySOT. For fair comparison, we follow the same training protocol (datasets and training hyper-parameters) for our SiamFC-ACM, SiamRPN++ACM and SiamBAN-ACM as that of their respective baseline SiamFC, SiamRPN++ and SiamBAN trackers. Further, we use the same loss functions in our tracking networks as that of the respective baselines, as ACM can be optimized without auxiliary guidance. We perform training on a workstation with an Intel E5-2698 v4 CPU, 512G memory, and four V100 GPUs. For both training and testing, template patches are cropped to $127 \times 127$ pixels, and the search region is cropped to $255 \times 255$ pixels.

### 4.1. State-of-the-Art Comparison

**TrackingNet [30]:** Table 1 shows the comparison on TrackingNet test set, which comprises over 500 videos without publicly available ground-truths. The results are obtained through an online evaluation server. Our three trackers (SiamFC-ACM, SiamRPN++ACM and SiamBAN-ACM) achieve consistent improvement over their respective baselines (SiamFC, SiamRPN++ and SiamBAN). The recently introduced KYS [4] and its baseline DiMP [3] achieve normalized precision (**NP**) scores of 80.0 and 80.1, respectively. Our SiamBAN-ACM achieves **NP** score of 81.0, outperforming both KYS and DiMP. SiamBAN-ACM also achieves favorable result in terms of success (**A**), against existing trackers with an AUC score of 75.3.

**OTB-100 [45]:** Fig.6(a) shows the results, in terms of success plot, over all 100 videos of OTB-100. The trackers are ranked in terms of their AUC score (in the legend). Among existing methods, SiamBAN achieves an AUC score of 69.6. The recently introduced KYS [4] and its baseline DiMP [3] obtain AUC scores of 69.4 and 68.8, respectively. Our SiamBAN-ACM outperforms existing trackers with an AUC score of 72.0. Further, our SiamBAN-ACM obtains an absolute gain of 2.5% over the baseline SiamBAN. In OTB-100, each video is annotated with 11 different attributes. SiamBAN-ACM achieves promising performance on all these attributes, compared to existing methods. The attribute plots are provided in the supplementary material.

**LaSOT [12]:** We evaluate our approach on the test set comprising 280 long videos. Fig.6(b) shows the success plot. We rank the trackers w.r.t. their AUC scores (in the legend). Among existing methods, SiamBAN and DiMP obtain AUC scores of 51.4 and 56.5, respectively. Our SiamBAN-ACM obtains favorable results against the state-of-the-art, while outperforming baseline SiamBAN by an AUC gain of 5%.

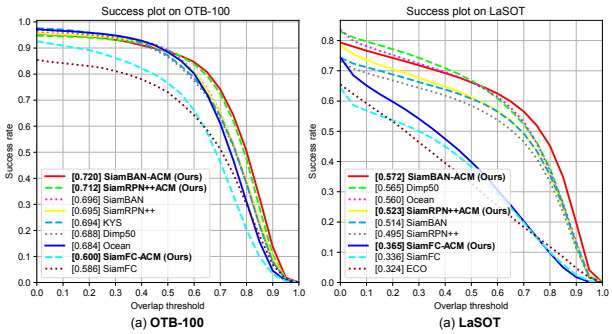


Figure 6. **State-of-the-art comparison on (a) OTB-100 [45] and (b) LaSOT [12] test set** in terms of success plot. For each method, we show the AUC scores in the legend. On both datasets, our ACM-based trackers (SiamFC-ACM, SiamRPN++ACM and SiamBAN-ACM) consistently outperform their respective baselines (SiamFC, SiamRPN++ andSiamBAN). Best viewed zoomed in.

| | SiamFC [40] | **SiamFC -ACM** | DiMP [3] | SiamRPN++ [25] | SiamBAN [5] | KYS [4] | **SiamRPN++ ACM** | **SiamBAN -ACM** |
|---|---|---|---|---|---|---|---|---|
| **A** | 0.571 | 0.577 | 0.740 | 0.733 | 0.725 | 0.740 | 0.747 | 0.753 |
| **P** | 0.553 | 0.537 | 0.687 | 0.694 | 0.687 | 0.688 | 0.705 | 0.712 |
| **NP** | 0.652 | 0.675 | 0.801 | 0.800 | 0.795 | 0.800 | 0.804 | 0.810 |

Table 1. **State-of-the-art comparison on TrackingNet [30] test set** in terms of success (AUC), precision and normalized precision. Success, precision and normalized precision are denoted by **A**, **P** and **NP**, respectively. The best two results are shown in red and blue, respectively.

| | SiamFC [2] | **SiamFC -ACM** | SiamRPN++ [25] | ROAM++ [49] | SPM [41] | **SiamRPN++ ACM** | SiamBAN [5] | **SiamBAN -ACM** |
|---|---|---|---|---|---|---|---|---|
| **E** | 0.277 | 0.338 | 0.441 | 0.434 | 0.481 | 0.501 | 0.505 | 0.549 |
| **R** | 0.382 | 0.294 | 0.174 | 0.210 | 0.206 | 0.144 | 0.149 | 0.098 |
| **A** | 0.549 | 0.535 | 0.599 | 0.620 | 0.610 | 0.666 | 0.632 | 0.647 |

Table 2. **State-of-the-art comparison on VOT2016 challenge dataset [23]** in terms of expected average overlap (**E**), robustness (**R**) and accuracy (**A**). The best two results are shown in red and blue fonts, respectively.

| | SiamFC [2] | **SiamFC -ACM** | SiamRPN++ [25] | **SiamRPN++ ACM** | DiMP [3] | SiamBAN [5] | Ocean [54] | **SiamBAN -ACM** |
|---|---|---|---|---|---|---|---|---|
| **E** | 0.163 | 0.206 | 0.285 | 0.303 | 0.321 | 0.327 | 0.350 | 0.362 |
| **R** | 0.958 | 0.712 | 0.482 | 0.431 | 0.371 | 0.396 | 0.316 | 0.316 |
| **A** | 0.470 | 0.503 | 0.599 | 0.624 | 0.582 | 0.602 | 0.594 | 0.621 |

Table 3. **State-of-the-art comparison on VOT2019 challenge dataset [23]** in terms of expected average overlap (**E**), robustness (**R**) and accuracy (**A**). The best two results are shown in red and blue fonts, respectively.

| SiamFC [2] | **SiamFC -ACM** | SiamRPN++ [25] | DiMP [3] | **SiamRPN++ ACM** | SiamCAR [13] | SiamBAN [5] | **SiamBAN -ACM** |
|---|---|---|---|---|---|---|---|
| 0.498 | 0.508 | 0.613 | 0.654 | 0.634 | 0.614 | 0.631 | 0.648 |

Table 4. **State-of-the-art comparison on UAV123 [29]** in terms of success (AUC). The best two results are shown in red and blue fonts, respectively.

**VOT 2016 and 2019 [23]:** Table 2 and 3 show a comparison on VOT 2016 and 2019, respectively. On VOT2016, our SiamBAN-ACM outperforms the previous best SiamBAN with a EAO (**E**) absolute gain of 4.4%. Similarly on VOT 2019, our three trackers (in bold) achieve consistent improvement in performance over their baselines. Compared to SiamBAN, our SiamBAN-ACM has 20% lower failure rate, while also achieving improved tracking accuracy.

**UAV123 [29]:** Table 4 shows the comparison in terms of success (AUC). Among existing Siamese trackers, SiamCAR and SiamBAN achieve AUC scores of 61.4 and 63.1, respectively. Our SiamBAN-ACM achieves favorable performance against existing trackers with AUC score of 64.8.

### 4.2. Ablation Study

We perform an ablation study to analyze the impact of ACM in the three tracking architectures. As discussed earlier, our ACM addresses the limitations of XCorr and DW-XCorr by introducing an asymmetric convolution (AC). Further, ACM also possesses the flexibility to incorporate additional (non-visual) information in the form of b-box size. Here, we also analyze the impact of only replacing the XCorr or DW-XCorr with AC and not incorporating additional (b-box size) prior information. We perform ablation experiments on the VOT2016 and OTB-100 datasets. We follow the standard evaluation protocols of the respective datasets. On VOT2016, trackers are evaluated using expected average overlap (EAO) score. The EAO score takes into account both robustness and accuracy. Here, robustness represents number of tracking failures, while accuracy indicates the average overlap between the ground-truth b-box and tracker prediction. On OTB-100, trackers are evaluated using the area-under-the-curve (AUC), which is obtained by averaging the overlap precision (OP) scores over a range of thresholds [0, 1]. Here, OP metric indicates the percentage of frames where intersection-over-union (IoU) overlap between the ground-truth b-box and predictions from the tracker is greater than a certain threshold.

Table 5 shows the results using three baseline tracking architectures on both datasets. We also report the speed in terms of frames per second (FPS). Note that all speeds are reported on a GTX1080Ti GPU. On VOT2016, the baseline SiamBAN and SiamRPN++ achieve EAO scores of 50.5 and 46.4, respectively. A consistent improvement in tracking performance is obtained when replacing the DW-XCorr with our AC in these two baseline architectures. Our final ACM, which contains both the AC and the prior b-box size information, achieves significant improvement in performance over both the baselines. Our ACM-based trackers (SiamBAN-ACM and SiamRPN++ACM) obtain absolute gains of 4.4% and 3.7%, in terms of EAO, over their respective SiamBAN and SiamRPN++ baselines. In case of the baseline SiamFC, our ACM contains only the AC and no additional (non-visual) information, since SiamFC only needs to predict the center of the target. Our ACM-based tracker (SiamFC-ACM) obtains a significant gain of 6.1% over the baseline SiamFC. Similarly, our ACM-based trackers also provide consistent improvements in tracking performance on their respective baselines on OTB-100.

| | AC | ACM | VOT2016 (EAO) | OTB2015 (AUC Score) | Speed (fps) |
|---|---|---|---|---|---|
| SiamBAN | | | 0.505 | 0.695 | 48 |
| | ✓ | | 0.535 | 0.715 | 41 |
| | ✓ | ✓ | **0.549** | **0.720** | 41 |
| SiamRPN++ | | | 0.464 | 0.695 | 46 |
| | ✓ | | 0.485 | 0.705 | 40 |
| | ✓ | ✓ | **0.501** | **0.712** | 40 |
| SiamFC | | | 0.277 | 0.586 | 190 |
| | ✓ | | **0.338** | **0.600** | 172 |

Table 5. **Ablation study on VOT2016 [22] and OTB-100 [45]**. We show the results using three different baseline tracking architectures. All speeds are reported on a GTX1080Ti GPU. We also show our ACM with only AC and without the integration of prior non-visual information. In all cases, our final ACM achieves consistent improvement in tracking performance over the baseline architectures. The best scores are highlighted in bold in each case.

## 5. Conclusion

We propose a learnable module, called the asymmetric convolution (ACM), to efficiently fuse feature maps of different sizes in Siamese trackers. Our ACM addresses the limitations of standard DW-XCorr and benefits from large-scale offline training. Further, ACM possesses the flexibility to integrate useful non-visual information, such as the location (b-box size) of target b-box in the initial frame. We integrate ACM into three Siamese tracking architectures. Experiments on six datasets demonstrate that ACM-based trackers provide consistent improvement over their baselines, leading to favorable results against existing methods. Also we believe ACM would benefit other tasks.

## References


[1] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *NeurIPS*, pages 523–531, 2016. 3

[2] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, pages 850–865, 2016. 1, 3, 6, 7, 8

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6182–6191, 2019. 3, 7, 8

[4] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Know your surroundings: Exploiting scene information for object tracking. *arXiv preprint arXiv:2003.11014*, 2020. 3, 7

[5] Zedu Chen, Bineng Zhong, Guorong Li, Shengping Zhang, and Rongrong Ji. Siamese box adaptive network for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6668–6677, 2020. 1, 3, 6, 7, 8

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *CVPR*, pages 4660–4669, 2019. 3

[7] Xingping Dong and Jianbing Shen. Triplet Loss in Siamese Network for Object Tracking. In *European Conference on Computer Vision*, pages 459–474, 2018. 3

[8] Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli. Hyperparameter Optimization for Tracking With Continuous Deep Q-Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 518–527, 2018. 3

[9] Xingping Dong, Jianbing Shen, Wenguan Wang, Ling Shao, Haibin Ling, and Fatih Porikli. Dynamical Hyperparameter Optimization via Deep Reinforcement Learning in Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, November 2019. 3

[10] Xingping Dong, Jianbing Shen, Dongming Wu, Kan Guo, Xiaogang Jin, and Fatih Porikli. Quadruplet Network With One-Shot Learning for Fast Visual Object Tracking. *IEEE Transactions on Image Processing*, 28(7):3516–3527, July 2019. 3

[11] Fei Du, Peng Liu, Wei Zhao, and Xianglong Tang. Correlation-guided attention for corner detection based visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6836–6845, 2020. 1

[12] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, pages 5374–5383, 2019. 2, 3, 7

[13] Dongyan Guo, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen. Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6269–6277, 2020. 1, 3, 8

[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. 3

[15] C.R. Harris and et al. Array programming with NumPy. *Nature*, 2020. 2, 5

[16] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4834–4843, 2018. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[18] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, pages 749–765, 2016. 3

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[20] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *Proceedings of the IEEE International Conference on Computer Vision*, page 105–114, 2017. 3

[21] Adam Kosiorek, Alex Bewley, and Ingmar Posner. Hierarchical attentive recurrent tracking. In *NeurIPS*, pages 3053–3061, 2017. 3

[22] Matej Kristan, Aleš Leonardis, Jiři Matas, Michael Felsberg, Roman Pflugfelder, and et al. The visual object tracking vot2016 challenge results. In *LNCS*, volume 9914, pages 777–823, 2016. 8

[23] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *TPAMI*, pages 2137–2155, 2016. 7, 8

[24] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010. 6

[25] 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*, pages 4282–4291, 2019. 2, 3, 4, 6, 7, 8

[26] 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. 1, 3

[27] Zhiyuan Liang and Jianbing Shen. Local semantic siamese networks for fast tracking. ieee trans. on image processing. *IEEE Trans. on Image Processing*, 29:3351–3364, 2020. 3

[28] Yuanpei Liu, Xingping Dong, Xiankai Lu, Fahad Shahbaz Khan, Jianbing Shen, and Steven Hoi. Teacher-Students Knowledge Distillation for Siamese Trackers. *arXiv:1907.10586 [cs]*, November 2019. 3

[29] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *ECCV*, pages 445–461, 2016. 7, 8

[30] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–317, 2018. 7

[31] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, pages 4293–4302, 2016. 3

[32] Eunbyung Park and Alexander C Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. In *ECCV*, pages 569–585, 2018. 3

[33] Shi Pu, Yibing Song, Chao Ma, Honggang Zhang, and Ming-Hsuan Yang. Deep attentive tracking via reciprocative learning. In *NeurIPS*, pages 1931–1941, 2018. 3

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, pages 91–99, 2015. 1, 3

[35] Jianbing Shen, Xin Tang, Xingping Dong, and Ling Shao. Visual object tracking by hierarchical attention siamese network. *IEEE Transactions on Cybernetics*, 50(7):3068–3080, 2019. 3

[36] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson W. H. Lau, and Ming-Hsuan Yang. CREST: convolutional residual learning for visual tracking. In *ICCV*, pages 2574–2583, 2017. 3

[37] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *CVPR*, pages 8990–8999, 2018. 3

[38] Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, Alban Desmaison, Alykhan Tejani, Andreas Kopf, James Bradbury, Luca Antiga, Martin Raison, Natalia Gimelshein, Sasank Chilamkurthy, Trevor Killeen, Lu Fang, and Junjie Bai. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 5

[39] Chong Sun, Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Correlation tracking via joint discrimination and reliability learning. In *CVPR*, pages 489–497, 2018. 3

[40] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, pages 1420–1429, 2016. 3, 7

[41] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. In *CVPR*, pages 3643–3652, 2019. 7

[42] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NeurIPS*, pages 809–817, 2013. 3

[43] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional siamese network for high

performance online visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4854–4863, 2018. 3

[44] Xiao Wang, Chenglong Li, Bin Luo, and Jin Tang. Sint++: Robust visual tracking via adversarial positive instance generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4864–4873, 2018. 3

[45] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 7, 8

[46] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 3

[47] Yinda Xu, Zeyu Wang, Zuoxin Li, Ye Yuan, and Gang Yu. Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. In *AAAI*, pages 12549–12556, 2020. 1, 3

[48] Tianyu Yang and Antoni B. Chan. Learning dynamic memory networks for object tracking. In *European Conference on Computer Vision*, pages 152–167, 2018. 3

[49] Tianyu Yang, Pengfei Xu, Runbo Hu, Hua Chai, and Antoni B Chan. Roam: Recurrently optimizing tracking model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6718–6727, 2020. 7

[50] Yuechen Yu, Yilei Xiong, Weilin Huang, and Matthew R Scott. Deformable siamese attention networks for visual object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6728–6737, 2020. 1

[51] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, pages 2711–2720, 2017. 3

[52] Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *European Conference on Computer Vision*, pages 351–366, 2018. 3

[53] Zhipeng Zhang and Houwen Peng. Deeper and wider siamese networks for real-time visual tracking. In *CVPR*, pages 4591–4600, 2019. 3

[54] Zhipeng Zhang and Houwen Peng. Ocean: Object-aware anchor-free tracking. *arXiv preprint arXiv:2006.10721*, 2020. 8