Siamese Attentional Keypoint Network for High Performance Visual Tracking

Peng Gao[a], Yipeng Ma[b], Ruyue Yuan[c], Liyi Xiao[a], Fei Wang[a]

[a]*Harbin Institute of Technology, Shenzhen*
[b]*Huawei Noahs Ark Lab*
[c]*HiSilicon Technologies Co., Ltd.*

Abstract

Visual tracking is one of the most fundamental topics of computer vision. Numerous tracking approaches based on either discriminative correlation filters or Siamese convolutional networks have attained remarkable performance over the past decade. However, it has still been commonly recognized as an open research problem to develop a robust and effective tracker that can achieve satisfactory performance with high computational and memory storage efficiency in real-world scenarios. In this paper, we investigate impacts of three main aspects of visual tracking, i.e., the backbone network, the attentional mechanism and the detection component, and propose a Siamese Attentional Keypoint Network, dubbed SATIN, to achieve efficient tracking and accurate localization. Firstly, a new Siamese lightweight hourglass network is specifically designed for visual tracking. It takes advantage of the benefits of the repeated bottom-up and top-down inference to capture more global and local contextual information at multiple scales. Secondly, a novel cross-attentional module is utilized to leverage both channel-wise and spatial intermediate attentional information, which enhance both discriminative and localization capabilities of feature maps. Thirdly, a keypoints detection approach is invented to track any target object by detecting the top-left corner point, the centroid point and the bottom-right corner point of its bounding box. To the best of our knowledge, we are the first to propose this approach. Therefore, our SATIN tracker not only has a strong capability to learn more effective object representations, but also computational and memory storage efficiency, either during the training or testing stage. Without bells and whistles, experimental results demonstrate that our approach achieves state-of-the-art performance on several recent benchmark datasets, at speeds far exceeding the frame-rate requirement.

Keywords: Visual tracking, Siamese hourglass networks, cross-attentional module, keypoint detection

1. Introduction

Visual tracking is an important research problem in the field of computer vision. It is a task that estimates the position of an arbitrary target in the video sequence, as long as the target object

Email address: pgao@stu.hit.edu.cn (Peng Gao)

is specified with a bounding box in the first frame. Especially, high performance visual tracking approaches with real-time speed are widely pursued in numerous applications, including video surveillance, autonomous driving, pose estimation and human-computer interfaces [1, 2, 3, 4]. Despite considerable progress has been made by many tracking approaches in the last decade, visual tracking is still a challenging problem due to multiple negative scenarios such as occlusions, fast motions, scale variations and background clutters.

Most existing tracking approaches are developed based on two successful frameworks. The first one is the discriminative correlation filter (DCF). Thanks to circular shifting of training samples and fast learning correlation filters in the Fourier frequency domain, DCF demonstrates superior computational efficiency and fairly good tracking accuracy, and has attracted increasing attention since it was first exploited by MOSSE [5]. Later, in order to overcome different kinds of challenges and achieve competitive tracking performance, the advancement in DCF-based tracking is focused on the use of kernel functions [6], motion information [7, 8], multi-dimensional features [9, 10], multi-scale estimation [11, 12], boundary effects alleviation [13, 14], deep convolutional features [15, 16] and ensemble combination [17, 18, 19].

However, most DCF-based trackers employ offline pre-trained convolutional neural networks (CNN) for feature extraction, and do not perform stochastic gradient descent (SGD) for online fine-tuning parameters, so that DCF-based trackers benefit very little from end-to-end trainable networks. In recent years, another popular visual tracking framework occurs, i.e., Siamese convolutional neural network (SiamNet), which has shown remarkable potential for high performance visual tracking. These SiamNet-based trackers use a Siamese convolutional network to compare an exemplar image with candidate images of the same size, and hence bypass online fine-tuning problem [20, 21, 22, 23]. Several recent extensions [24, 25] exploit a SiamNet for feature extraction and a Region Proposal Networks (RPN) [26, 27] for classification and detection, have yielded excellent results on various benchmark datasets.

Unfortunately, the visual tracking community is still plagued by several problems. First, existing backbone networks [28, 29, 30] applied to visual tracking are always pre-trained for other different tasks such as ImageNet classification and detection [31], which are not adequate for effective tracking. In these networks, feature maps extracted from high-level convolutional layers always have low resolutions which are not sufficient for accurate localization, while high resolution feature maps extracted from lower layers are ineffective to distinguish targets with different attributes or categories. Moreover, these pre-trained networks are not lightweight enough for real-world applications. Second, it is well known that intermediate relationships of different convolutional layers demonstrate true potential for improving the representation and generalization capability [32, 33], but only a few approaches take full use of these relationships efficiently in visual tracking. Third, in order to obtain more accurate tracking results, recent trackers [24, 25] employ RPN for proposing target candidates. However, RPN always requires a large number of anchor boxes [26, 27] with redundant hyperparameters, i.e. the number, scales and aspect ratios of the candidate proposals, which incur computational and memory storage overload and make these approaches not efficient in both training and testing stages.

To address above issues, we develop the *Siamese Attentional Keypoint Network*, dubbed SATIN, for high performance visual tracking in an end-to-end fashion. It consists of three components, i.e., a Siamese lightweight hourglass network for feature extraction, a cross-attentional module for

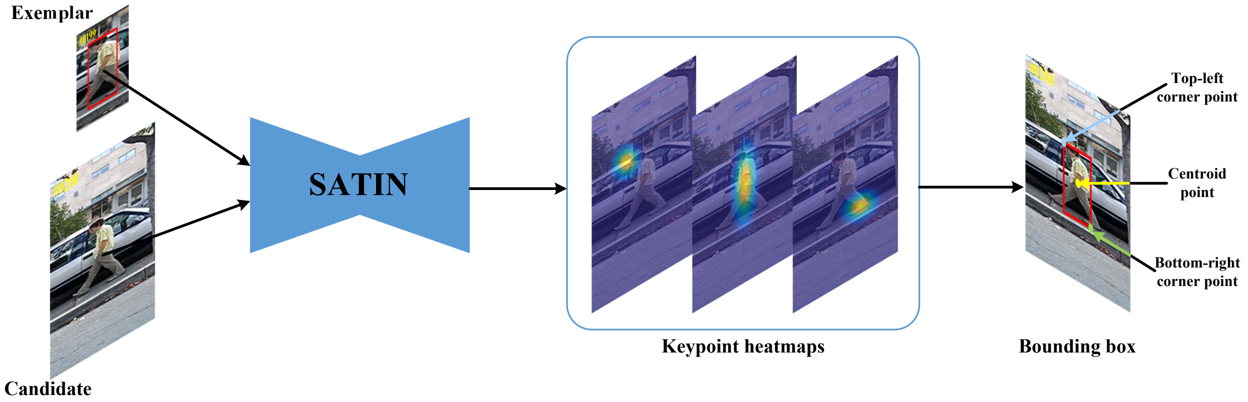

Figure 1: Visualization of heatmaps of SATIN. Tracking result on the 200th frame of the sequence *David3* is presented, with the top-left corner point, the centroid point and the bottom-right corner point marked.

adaptive feature refinement and a keypoints detection module for object detection and localization. As the backbone network of SATIN, the Siamese lightweight hourglass network is symmetric, consisting of (a) a series of convolutional layers for feature map resolution reductions, (b) a series of upsampling layers to raise the low resolution feature maps to the original resolution and (c) a set of skip layers to bring back original information. Therefore, our backbone network takes advantage of the benefits of the repeated bottom-up and top-down inferences, and captures more contextual information at multiple scales on the input images. Moreover, to boost the representation power of SATIN, we investigate the impact of spatial and channel-wise attention mechanisms. It is well known that visual attention plays the most important role in many computer vision tasks. We align our motivation to the recent convolutional block attention module (CBAM) [33], and develop a novel cross-attentional module consisting of a multi-layer perceptron and a single-layer perceptron to highlight informative representations and suppress the background noises. Since the low-level geometric information is effective for target localization and the high-level semantic information is benefit for target discrimination, the spatial and channel-wise attention maps can be computed separately on different level information. Further, motivated by CornerNet [34], we propose a unified detection architecture, which tracks target object using a set of keypoints, as shown in Fig. 1. It detects the top-left corner point, the centroid point and the bottom-right corner point of the target bounding box using three separate detection module. The centroid point detection module is similar to SiamFC [21]. Each corner point detection module has a localization branch and an offset prediction branch, starting with its own corresponding pooling layer, namely, *corner pooling* [34], to overcome the limitation of receptive field and enhance the capability to localize corner points of the bounding box. These corner points are determined by the heatmaps, and their locations are adjusted by the offsets. Compared with recent RPN methods, our keypoints detection module is simple, which eliminates the need for designing anchor boxes, and therefore significantly reduces the computational and memory storage overhead.

Our main contributions can be summarized as three folds. (a) We design a lightweight hourglass network as the backbone network which can capture more contextual information at multiple scales on the input images. (b) We develop a novel cross-attentional module to selectively highlight meaningful information and boost the representation power of feature maps. (c) We track the object by detecting the top-left corner, the centroid and the bottom-right corner of the target bounding box to eliminate design anchor boxes or multi-scale pyramids. Finally, our proposed approach achieves consummate performance both in terms of accuracy and robustness on several recent benchmark datasets [35, 36, 37, 38]. To the best of our knowledge, we are the first to formulate visual tracking as a keypoints detection task.

The rest of the paper is organized as follows. Section 2 briefly reviews related works. Section 3 illustrates the proposed SATIN tracker. Section 4 details experiments and discusses results. Finally, Section 5 concludes this paper with final remarks.

2. Related works

In this section, we give a brief review on three aspects related to our work: backbone networks, attentional mechanisms and detection components.

2.1. Backbone networks

In recent years, CNNs have made great progress in a wide range of computer vision applications due to their impressive representation abilities. Because of their surprisingly good performance of CNN on object classification and detection tasks, researchers are encouraged to either combine existing CNNs with DCF or design deep networks in a Siamese framework for high performance visual tracking. The most popular backbone networks utilized in recent trackers [16, 39, 40, 21, 24, 8] are AlexNet [28], VGGNet [29] and ResNet [30]. AlexNet [28] consists of several convolutional and pooling layers, and it was the first large-scale CNN that had won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [31]. VGGNet [29] stacks numerous small convolutional kernels without using pooling operations, which increases the representation power of the network while reducing the number of parameters. ResNet [30] introduces the skip connection to learn residual information, which makes it more efficient and simple to design deeper architectures.

Moreover, there are some efficient backbone networks introduced by other vision tasks, such as hourglass networks [41] and FlowNet [42]. However, these networks sometimes are computational and memory expensive for practical computer vision applications. Meanwhile, some lightweight networks [43, 44, 45] focus on designing more efficient architecture to reduce network computation while maintaining excellent performance. Unfortunately, these networks mentioned above are always pre-trained for object classification and detection. Trackers that employ these networks may obtain suboptimal tracking results. The recent trend in visual tracking [46, 22, 21] is to design suitable networks for learning object- or domain-specific representations and enhance the generalization power to new video sequences. Different from these tracking methods, we aim to design a lightweight CNN as the backbone network, which can learn more contextual features at multiple scales with a simple architecture and fewer parameters.

2.2. Attentional mechanisms

Attentional mechanisms have been utilized by many computer vision tasks, such as image classification [32], object detection [33] and image semantic segmentation [47]. The recent advance of tracking approaches has achieved great success by integrating attentional mechanisms.

CSRDCF [48] introduces a channel weighted block by constructing a unique spatial reliability map in an online constrained optimization. LSART [49] proposes a cross-patch similarity kernel to encode the similarity scores of all pairs of patches between two samples, and a spatial regularized constraint method to enforce each filter kernel to focus on a specific region of the target. Both RASNet [50] and FlowTrack [7] adopt a squeeze-and-excitation network (SENet) [32] to enhance the representation capability of their backbone networks. In these modules, global average-pooling operations are utilized to compute channel-wise attention and to re-weight the original feature maps based on the pooled features. In contrast to those attentional mechanisms, we design a cross-attentional module in an end-to-end offline learning framework to investigate both spatial and channel-wise relationship between shallow and deep convolutional features, especially their geometric and semantic contributions to global contextual information.

2.3. Detection components

Most state-of-the-art tracking approaches follow the tracking-by-detection paradigm [51, 11, 39, 24] which treats visual object tracking as detection problems. DCF-based trackers [10, 11] and some SiamNet-based trackers [22] detect objects by searching the peak value of similarity score maps, which come from the correlation of the candidate feature maps with correlation filters trained on the exemplar feature maps. While other SiamNet-based trackers [21] straightforwardly cross-correlate exemplar and candidate feature maps. To achieve more accurate tracking in scale variation scenarios, most existing tracking approaches utilize multi-scale pyramids as DSST [11] and SAMF [12]. But these improvements in accuracy come at expensive cost of significant reductions in tracking speed. SiamRPN [24] and DaSiamRPN [25] combine RPN [27] with SiamNet to formulate the multiple scale object tracking as proposal classification and bounding box regression problem on correlation feature maps. However, these trackers always require a large number of anchor boxes, and many extra hyperparameters need to be set in advance to generate more suitable proposals. Unlike these detection methods, and motivated by object detection approach in CornerNet [34], we track an object as a set of keypoints, i.e. the top-left corner point, the centroid point and the bottom-right corner point of the bounding box, therefore mitigate the drawbacks of the above-mentioned detection components.

3. The proposed approach

3.1. Algorithm overview

In this section, we describe our proposed SATIN in detail. Fig. 2 shows the overall framework of SATIN, which consists of three portions: a backbone network for deep feature extraction, a cross-attentional module for representation capability improvement and a keypoints detection module for keypoints prediction. Given an exemplar image $\mathbf{z}$ centered on the target object and a larger candidate image $\mathbf{x}$ centered on the previous target location, these two images are fed into a backbone network consisting of two lightweight hourglass networks. Each lightweight hourglass network is designed with a symmetric topology that captures and consolidates both global and local information at multiple scales of the input image through repeated bottom-up and top-down processing. The architecture of our backbone network will be presented in Section 3.2. The backbone network is followed by a cross-attentional module which is consisted of a multi-layer

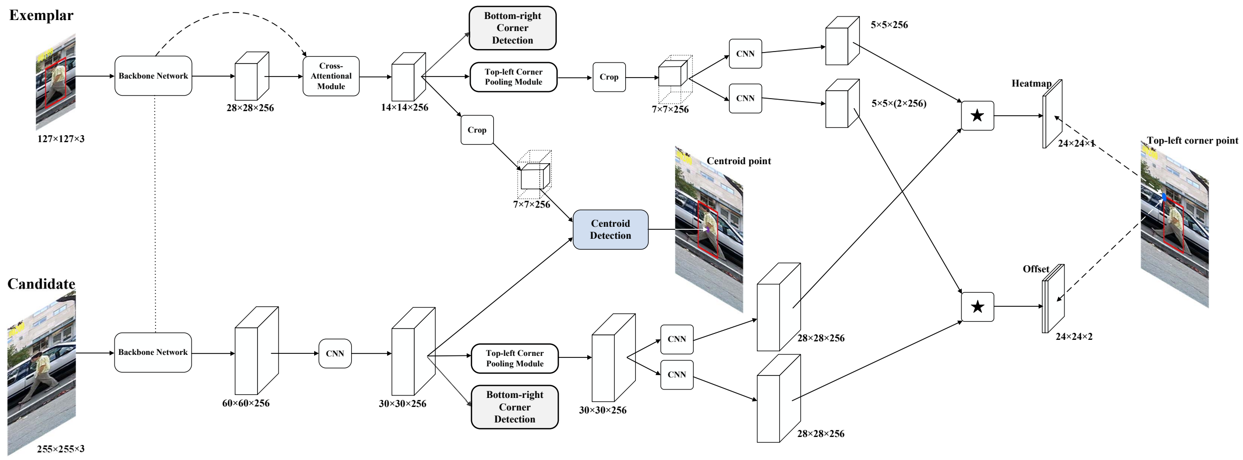

Figure 2: The overall framework of our proposed approach. For the sake of clarity, we only show pipelines of the top-left corner (bottom-right corner) point and the centroid point detection.

perceptron and a single-layer perceptron to highlight meaningful information along spatial and channel-wise dimensions, respectively. The spatial attention map is inferred from the low-level geometric features before being fed into the second lightweight hourglass network. While the channel-wise attention map is obtained from the high-level semantic feature map, which is the output of the second lightweight hourglass network. Both spatial and channel-wise attention maps are then multiplied with the output of the backbone network, therefore taking full advantage of the essential attention during tracking. We will illustrate details of our cross-attentional module in Section 3.3. After that, we exploit keypoints detection module to track target objects, which detects the top-left corner point, the centroid point and bottom-right corner point of the bounding box simultaneously. Each corner point detection module has its own corner pooling strategy, and correlates the paired exemplar feature map and the candidate feature map to obtain a heatmap for corner position prediction and a set of offsets. These offsets are then be used for corner location refinement. The bottom-right corner and the top-left corner are detected in a similar way. The centroid point is detected in the similar way to SiamFC [21]. For the sake of clarity, Fig. 2 demonstrates the processes of top-left corner point detection and centroid point detections. We will describe the implementation of the keypoint detection module in Section 3.4.

3.2. Lightweight hourglass network

It is well acknowledged that powerful object representations are crucial for accurate and robust visual tracking [52, 39]. Deep features extracted from the off-the-shelf CNNs pre-trained for other visual tasks may be suboptimal for visual tracking, there is still a considerable gap to achieve state-of-the-art tracking performance. Different from CNNs employed in most existing tracking approaches [21, 24, 39], we employ a Siamese lightweight hourglass network as the backbone network of SATIN, which is technically designed for visual tracking.

The hourglass network was first introduced into the computer vision community for human pose prediction [41]. In a conventional hourglass network, several *hourglass modules* are stacked

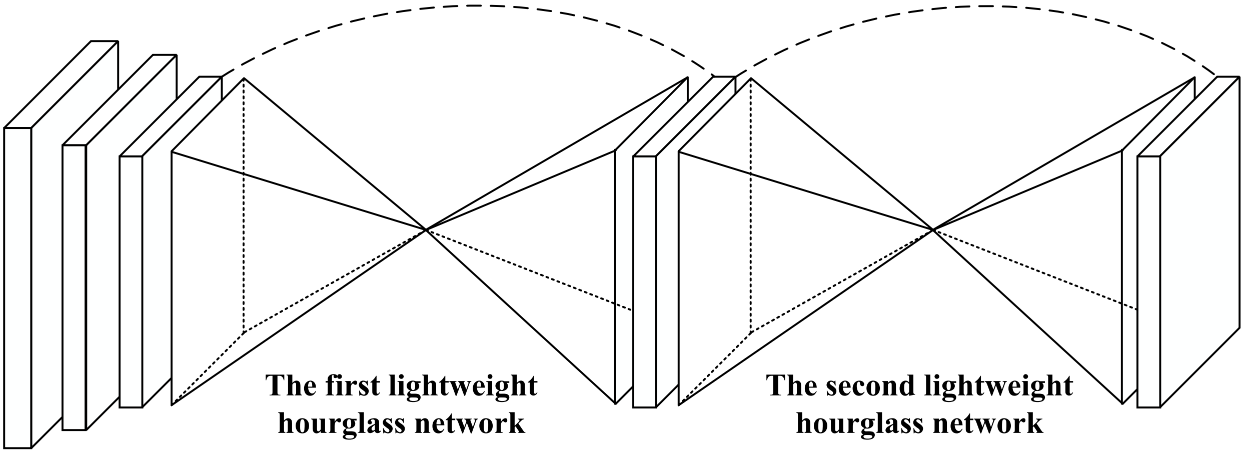
The first lightweight hourglass network
The second lightweight hourglass network

Figure 3: The architecture of our backbone network.

to preserve low-level geometric representation and high-level semantic information across different resolutions in the repeated bottom-up (from fine to coarse) and top-down (from coarse to fine) processing. Each hourglass module consists of several *residual blocks*, which are the minimal units in the network. The hourglass module first downsamples input features to coarse resolution using a series of convolutional and max-pooling layers to extract semantic information. The pooled features are then upsampled back to the original input resolution through a series of convolutional and upsampling layers. However, some useful fine-grained object-specific details may be lost during bottom-up processing. To solve this problem, a series of convolutional and skipping layers are utilized to bring the lost details back to the upsampled features. The upsampled features and lost details are combined across multiple scales as the inputs of the next module. The hourglass network takes advantage of both global and local contextual information throughout the repeated bottom-up and top-down framework. It is an ideal backbone network for deep features extraction for visual tracking. In our approach, two hourglass networks are stacked in the backbone network, as shown in Fig. 3.

In order to extract more powerful deep features, more hourglass modules are nested to make a deeper hourglass network. However, as the hourglass network going deeper, it becomes more complicated and not practical for real-world applications due to limited computational resources. Inspired by recent implementation of efficient and lightweight networks, we exploit the compression method to make some critical modifications to the conventional hourglass network and design the architecture of a lightweight hourglass network. Instead of using the 3×3 standard convolution layer to simultaneously filter and consolidate input features, we follow the depthwise separable convolutional unit described in MobileNet [43] and factorize the 3×3 standard convolution into a 3×3 depthwise convolution for filtering and a 1×1 pointwise convolution for consolidating. In the residual block, we use a depthwise separable convolutional unit followed by a depthwise convolutional layer to extract deep features, and another pointwise convolutional layer is used to bring the original information back to the output features.

In the hourglass module, three residual blocks are followed by an upsampling layer as the

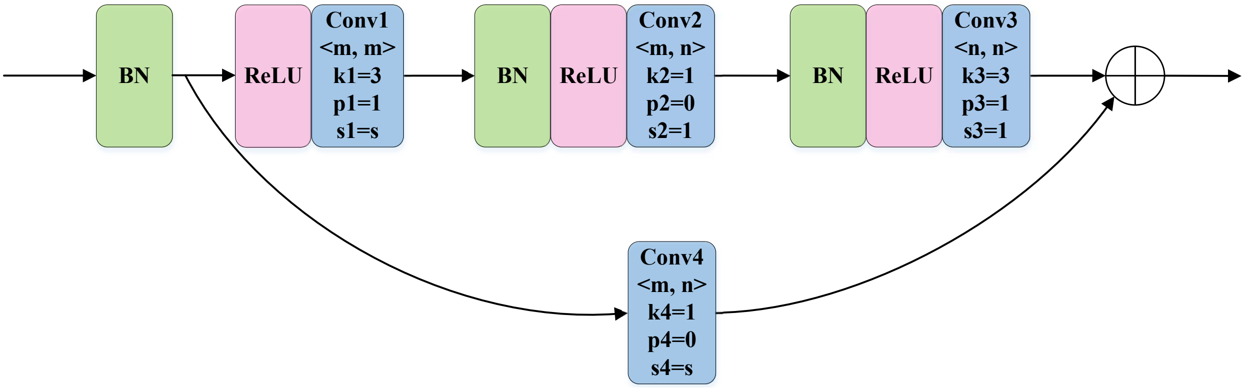

Figure 4: Illustration of a single residual block. In the figure, *BN* means the batch normalization, *ReLU* means a rectified linear unit and *Conv∗* means a convolutional layer. The first number in the bracket indicates the input channel and the second one indicates the output channel. The parameters $k*$, $p*$ and $s*$ represent the filter size, padding size and stride of the corresponding convolutional layer, respectively. The parameter s indicates the stride of Conv1 and Conv4, which is preset according to the hourglass module.

main propagation, and another residual block works as the skipping connection. It is worth noting that the second residual block M2 can be nested by an hourglass module, and thus the network can do repeated bottom-up and top-down processing. Four hourglass modules are nested with output channel numbers of $(256, 512, 512, 256)$ respectively in each hourglass network. Another noteworthy modification is we apply stride 2 instead of max-pooling operation in the first residual block of each hourglass module to reduce input feature resolution. A simple nearest neighbor upsampling method is used to restore output feature maps to the original input resolution. After that, the upsampled features and lost details are merged directly by element-wise addition. In addition, before the first hourglass network, we apply an 11×11 convolutional layer with stride 2 and output channel 128 and a 5×5 convolutional layer with stride 2 and output channel 256 to increase the number of input channels and reduce the input resolution. Fig. 4 and Fig. 5 show implementations of our residual block and hourglass module, respectively.

3.3. Cross-attentional module

The backbone network as designed in Section 3.2 is used to generate contextual representations of both exemplar image $\mathbf{z}$ and candidate image $\mathbf{x}$. However, those contextual representations may not be sufficient enough for tracking performance improvement, because they treat every channel and region equally. Some informative channels and attentive regions of the output are more crucial for target discrimination and localization than others, as they may involve more semantic attributes and visual patterns of different objects. In order to highlight meaningful information and to enhance the representation power of output features, an effective cross-attentional module in our approach is exploited as following.

Without loss of generality, the input and output feature maps of the second lightweight hourglass network are respectively denoted as $\mathbf{F}^i \in \mathbb{R}^{W \times H \times C}$ and $\mathbf{F}^o \in \mathbb{R}^{W \times H \times C}$, where W, H and C indicate the width, height and channel number of feature maps, respectively. Since low-level geometric representations contribute more to localizing the target, and the high-level semantic infor-

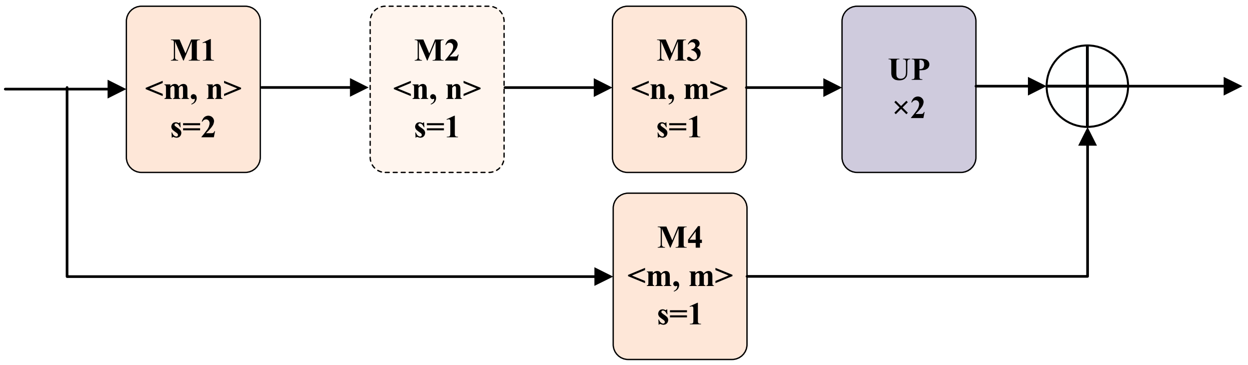

Figure 5: Illustration of a single hourglass module. In the figure, $M*$ means a residual block, UP means a upsampling layer. The first number in the bracket indicates the input channel and the second one indicates the output channel. The parameter s represents the stride of Conv1 and Conv4 in each residual block.

mation is great benefit for distinguishing the target from the background, we utilize a single-layer perceptron on $\mathbf{F}^i$ and a multi-layer perceptron on $\mathbf{F}^o$, therefore encode those regions and channels to be emphasized or suppressed, respectively. The structure of the cross-attentional module is shown in Fig. 6.

Spatial attention. We utilize spatial attention to highlight informative regions which effectively represent the current target object. Given the input feature map $\mathbf{F}^i \in \mathbb{R}^{W\times H\times C}$, we first apply global average-pooling and max-pooling operations along the channel axis to construct two 2D spatial feature descriptors, denoted as $\mathbf{F}^i_{avg} \in \mathbb{R}^{W\times H\times 1}$ and $\mathbf{F}^i_{max} \in \mathbb{R}^{W\times H\times 1}$. Spatial feature descriptors are then concatenated and fed into a single-layer perceptron with a sigmoid activation, where a 2D spatial attention map $\mathbf{M}_s \in \mathbb{R}^{W\times H\times 1}$ is created over the size of $W \times H$,

$$\mathbf{M}_s = \sigma(\texttt{SLP}([\mathbf{F}^i_{avg}; \mathbf{F}^i_{max}])) \tag{1}$$

where σ indicates the sigmoid function, and $[\mathbf{F}^i_{avg}; \mathbf{F}^i_{max}] \in \mathbb{R}^{W\times H\times 2}$ denotes the concatenation of global average-pooled and max-pooled feature descriptors. SLP means the single-layer perceptron consisting of a 7×7 convolutional layer with padding 3, stride 1, input channel 2 and output channel 1.

Channel-wise attention. The channel-wise attention is employed to adaptively select useful channels which facilitate the current tracking task. Given the output feature map $\mathbf{F}^o \in \mathbb{R}^{W\times H\times C}$, global average-pooling and max-pooling operations are applied along the spatial axis to generate two 1D channel-wise feature descriptors denoted as $\mathbf{F}^o_{avg} \in \mathbb{R}^{1\times 1\times C}$ and $\mathbf{F}^o_{max} \in \mathbb{R}^{1\times 1\times C}$, respectively. Different from the spatial-wise attention, a multi-layer perceptron is applied on each pooled feature descriptor to create a 1D channel attention map $\mathbf{M}_c \in \mathbb{R}^{1\times 1\times C}$ over C channels,

$$\begin{aligned}\mathbf{M}_c &= \sigma(\texttt{MLP}(\mathbf{F}^o_{avg}) \oplus \texttt{MLP}(\mathbf{F}^o_{max})) \\ &= \sigma\Big(W_u(W_d(\mathbf{F}^o_{avg})) \oplus W_u(W_d(\mathbf{F}^o_{max}))\Big)\end{aligned} \tag{2}$$

where σ indicates the sigmoid function, $\oplus$ denotes the element-wise addition, and MLP means the

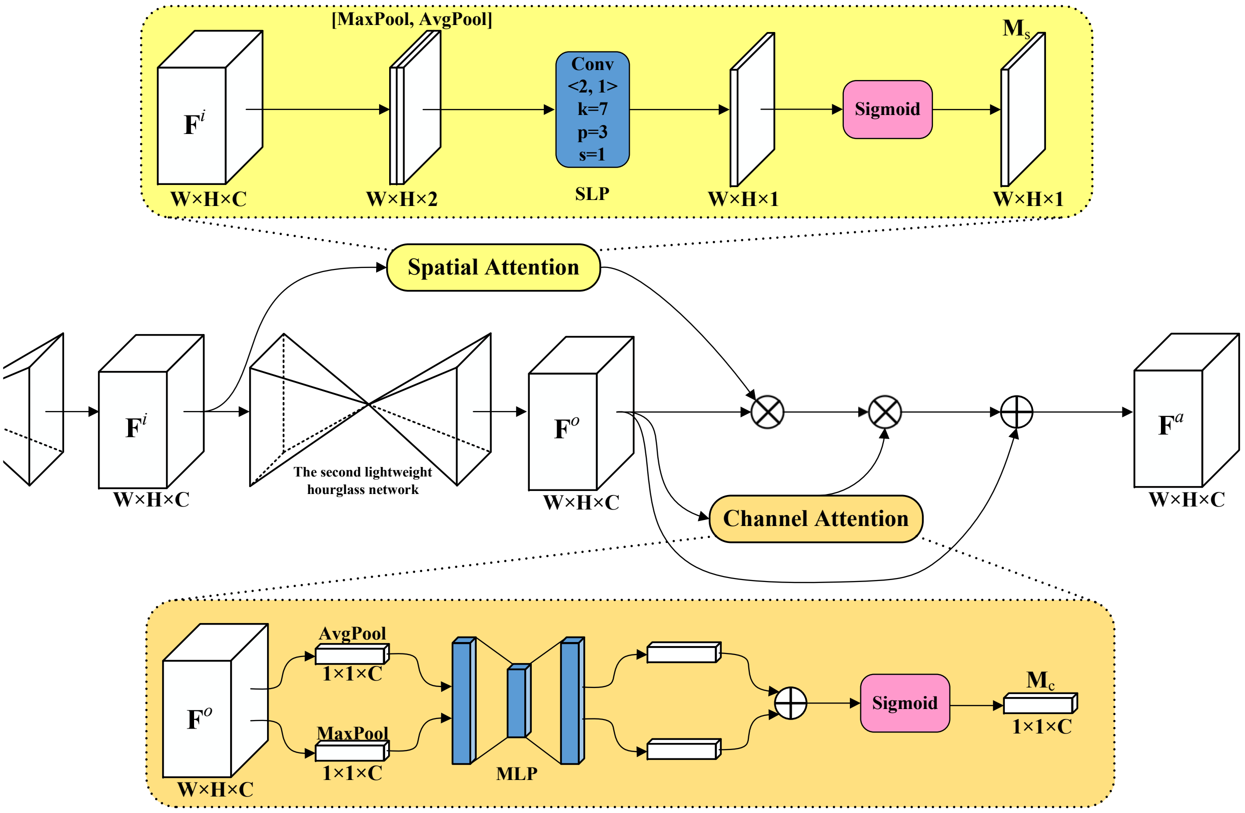

Figure 6: The structure of the cross-attentional module. The concatenation of max-pooled and average-pooled input feature maps are fed into a single-layer perceptron to infer a 2D spatial attention map. Both max-pooled and average-pooled output feature maps are fed into a shared multi-layer perceptron to generate a 1D channel-wise attention map. Finally, the output feature map is weighted with both channel-wise and spatial attention by broadcasting element-wise multiplication.

multi-layer perceptron. Moreover, the multi-layer perceptron is composed of a channel reduction layer with weight $W_d \in \mathbb{R}^{\frac{C}{r} \times C}$ and a ReLU activation, and a channel increasing layer with weight $W_u \in \mathbb{R}^{C \times \frac{C}{r}}$ and a sigmoid activation, where r (set to 4) is a channel reduction ratio [33] to reduce computational burden. It is worth noting that the weights W_d and W_u of the multi-layer perceptron are shared for both global average-pooled and max-pooled feature descriptors.

The attention-refined feature map $\mathbf{F}^a \in \mathbb{R}^{W \times H \times C}$ weighted with both channel-wise and spatial attention can be calculated sequentially as,

$$\mathbf{F}^a = \mathbf{F}^o \oplus (\mathbf{F}^o \otimes \mathbf{M}_s \otimes \mathbf{M}_c) \tag{3}$$

where $\otimes$ and $\oplus$ indicate element-wise addition and multiplication, respectively. Another noteworthy issue is that the spatial attention map $\mathbf{M}_s$ and the channel attention map $\mathbf{M}_c$ are broadcasted along the channel and the spatial axes respectively during element-wise multiplication. Original output information is brought back to the attention-refined feature map in the cross-attentional module.

In our approach, the cross-attentional module is only applied to the exemplar image. Before the keypoints detection process, we reduce the resolution of $\mathbf{F}^a(\mathbf{z})$ and $\mathbf{F}^o(\mathbf{x})$ by two times using 3×3 convolutional layers both with stride 2 and padding 1. For convenience, final feature maps of the exemplar image $\mathbf{z}$ and the candidate image $\mathbf{x}$ are denoted as $\mathbf{F}(\mathbf{z})$ and $\mathbf{F}(\mathbf{x})$, respectively.

3.4. Keypoint detection

We now show how those three keypoint detection modules work, i.e., the top-left corner point, the centroid point and the bottom-right corner point detection module.

Corner detection. The corner point detection modules are inspired by CornerNet [34] which detects the object as a pair of bounding box corner points to eliminate the need for designing anchor boxes [27].

There are two corner point detection modules in SATIN, one for the top-left corner point detection and the other for the bottom-right corner point detection. However, because of the limitation of the receptive field and the lack of local visual pattern of corner points, it is difficult to detect the corner point outside the target object directly, as shown in Fig. 1. To address this problem, we adopt a new type of pooling layer, namely, corner pooling [34], to translate meaningful object-specific information to locations outside of the target object based on explicit prior knowledge.

We use an example to illustrate our idea. Suppose we are going to detect the top-left corner point, as shown in Fig. 7, max-pooling operation is applied on the feature map horizontally from the rightmost boundary to the left, and vertically from the bottommost boundary to the top. For convenience, let $\mathbf{F} \in \mathbb{R}^{W \times H \times C}$ denote either $\mathbf{F}(\mathbf{z})$ or $\mathbf{F}(\mathbf{x})$, where W, H and C indicate the width, height and channel number of $\mathbf{F}$ respectively. The top-left corner point detection module first generates two corner-oriented feature maps $\mathbf{F}_t \in \mathbb{R}^{W \times H \times C}$ and $\mathbf{F}_l \in \mathbb{R}^{W \times H \times C}$ using two independent 3×3 convolutional layers with stride 1 and padding 1, respectively. Then, we apply max-pooling operation between coordinate (m, n) and (m, H) on $\mathbf{F}_t$ to obtain a vertical maximum pattern $t_{(m,n)}$, where $m \in \{1, \dots, W\}$ and $n \in \{1, \dots, H\}$. Similarly, the same operation is applied between (m, n) and (W, n) on $\mathbf{F}_l$ to obtain a horizontal maximum pattern $l_{(m,n)}$. The top-left feature pattern $tl_{(m,n)}$ at location (m, n) in the top-left corner-pooled feature map $\mathbf{F}_{tlc}$ is calculated by adding $t_{(m,n)}$ and $l_{(m,n)}$ together. The process is depicted as Eq. 4,

$$\begin{aligned} t_{(m,n)} &= \begin{cases} \max(\mathbf{F}_{t_{(m,n)}}, t_{(m,n+1)}) & \text{if } n < H \\ F_{t_{(m,H)}} & \text{otherwise} \end{cases} \\ l_{(m,n)} &= \begin{cases} \max(\mathbf{F}_{l_{(m,n)}}, l_{(m+1,n)}) & \text{if } m < W \\ F_{l_{(W,n)}} & \text{otherwise} \end{cases} \\ tl_{(m,n)} &= t_{(m,n)} + l_{(m,n)} \end{aligned} \tag{4}$$

where $\mathbf{F}_{t_{(m,n)}}$ (or $\mathbf{F}_{l_{(m,n)}}$) denote oriented feature patterns at location (m, n) in $\mathbf{F}_t$ (or $\mathbf{F}_l$). To obtain the top-left corner-specific feature map $\mathbf{F}_{tl}$, the top-left corner-pooled feature map $\mathbf{F}_{tlc}$ is then fed into a 3×3 convolutional layer, and add back the residual feature map $\mathbf{F}_{tlr}$ obtained by applying a 1×1 convolution to the input feature map $\mathbf{F}$.

Each corner detection module is utilized to detect only one corner point, and hence it only outputs a single-channel heatmap for prediction and a double-channel offset map for adjustments. To suppress the effect of redundant information, the top-left quarter of $\mathbf{F}_{tl}(\mathbf{z})$ is then cropped and

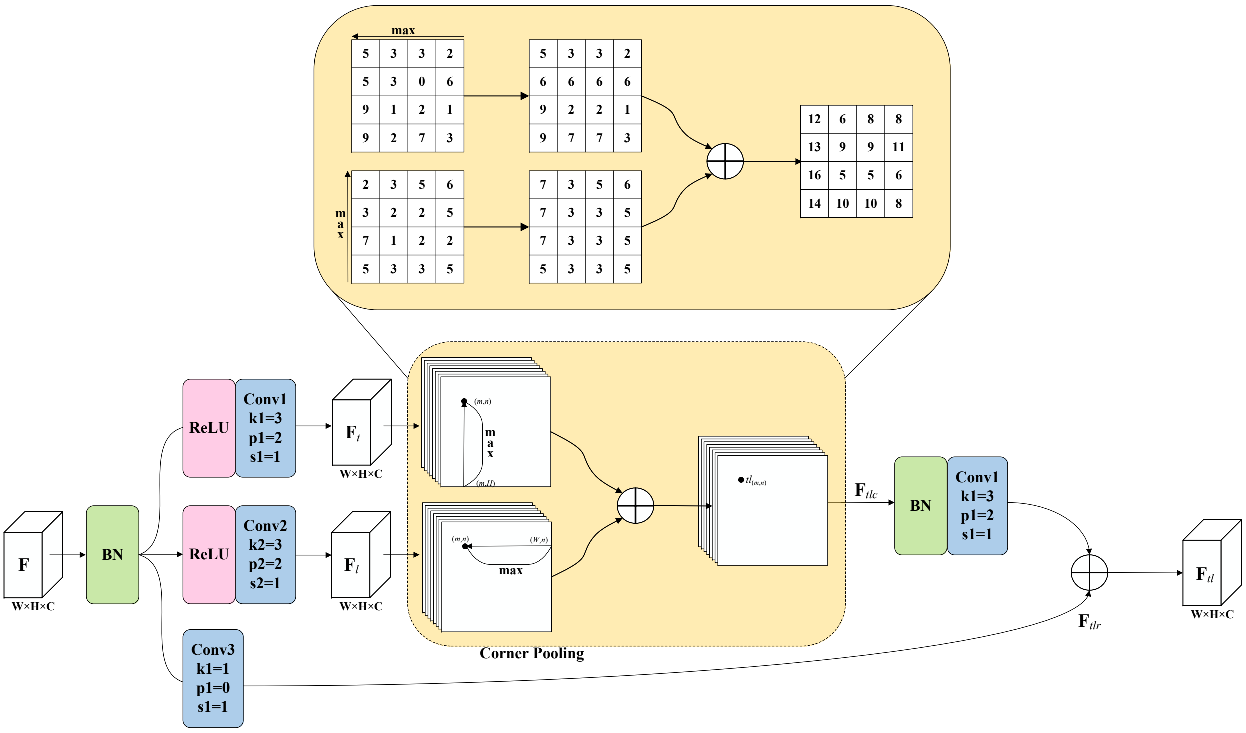

Figure 7: An illustration of the corner-pooling operation in the top-left corner point detection module.

denoted as $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]$. Therefore, two separate 3×3 convolutional layers are applied to keep and double the channels of $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]$, generating $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_s$ (with same number of channels) and $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_o$ (with doubled number of channels), respectively. Meanwhile, $\mathbf{F}_{tl}(\mathbf{x})$ is also divided into two branches $[\mathbf{F}_{tl}(\mathbf{x})]_s$ and $[\mathbf{F}_{tl}(\mathbf{x})]_o$ by two separate 3×3 convolutional layers but the channels are kept unchanged. The heatmap $\mathbf{S}_{lt}$ and offset $\mathbf{O}_{lt}$ are computed as,

$$\begin{aligned} \mathbf{S}_{lt} &= [\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_s * [\mathbf{F}_{tl}(\mathbf{x})]_s \\ \mathbf{O}_{lt} &= [\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_o * [\mathbf{F}_{tl}(\mathbf{x})]_o \end{aligned} \tag{5}$$

where $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_s$ and $[\texttt{crop}_{tl}(\mathbf{F}_{tl}(\mathbf{z}))]_o$ are served as correlation kernels, and $*$ denotes the cross-correlation operation. Each score in $\mathbf{S}_{lt}$ represents the probability of being the top-left corner point at location (m, n). After upsampling $\mathbf{S}_{lt}$ to the resolution of $\mathbf{x}$, the location of the maximum score relative to the top-left corner point of the bounding box.

Due to a series of downsampling operations involved in the previous modules, the top-left corner point (m_{lt}, n_{lt}) in $\mathbf{x}$ is mapped to the location $(\lfloor \frac{m_{lt}}{\alpha} \rfloor, \lfloor \frac{n_{lt}}{\alpha} \rfloor)$ in $\mathbf{S}_{lt}$, where $\lfloor \cdot \rfloor$ denotes the round down operation and α is the downsampling factor. However, if we remap the location $(\lfloor \frac{m_{lt}}{\alpha} \rfloor, \lfloor \frac{n_{lt}}{\alpha} \rfloor)$ from $\mathbf{S}_{lt}$ to $\mathbf{x}$ by multiplying α directly, we may not obtain its corresponding location (m_{lt}, n_{lt}), which significantly affects the tracking accuracy, especially when tracking small objects. To address this issue, we exploit the offset map $\mathbf{O}_{lt}$ to adjust the score locations before remapping them to the candidate image,

$$(m_{lt},\, n_{lt}) = \alpha \cdot \left(\left\lfloor \frac{m_{lt}}{\alpha} \right\rfloor + \mathbf{O}^1_{lt},\, \left\lfloor \frac{n_{lt}}{\alpha} \right\rfloor + \mathbf{O}^2_{lt} \right) \tag{6}$$

where $\mathbf{O}^a_{lt}$ indicates the a-th channel of $\mathbf{O}_{lt}$.

Similar procedure can be extended to the bottom-right corner point detection, but along the topmost boundary to the bottom and the leftmost boundary to the right. Two corner-oriented feature maps $\mathbf{F}_b$ and $\mathbf{F}_r$ are first generated respectively by applying two separate 3×3 convolutional layers on the feature map $\mathbf{F}$. The bottom-right corner pooling operation is then applied to create the feature map $\mathbf{F}_{br}$. The heatmap $\mathbf{S}_{br}$ and the offset map $\mathbf{O}_{br}$ are computed with formulation like Eq. 5.

Centroid detection. The centroid point detection module is just a variant of the corner point detection module, but with the corner pooling operation removed. To detect the centroid point of the bounding box, only a quarter around the center of the exemplar feature map $\mathbf{F}(\mathbf{z})$ is cropped so as to reduce the effect of background information, as illustrated in Fig. 2. After that, the heatmap $\mathbf{S}_c$ and the offset map $\mathbf{O}_c$ are created using similar correlation operations as described in Eq. 5.

3.5. Loss function

During training, it is conventional to assign a binary label $y \in \{0, 1\}$ for each pixel (m, n) in the candidate image $\mathbf{x}$ to indicate whether it is a keypoint or not. Instead of doing this, a soft label is exploited to penalize negative locations smoothly within a radius of the ground-truth keypoint. This is because for each keypoint, there is only one positive location while all other locations are negative. However, false keypoints sometimes detected close to their respective positive locations, but the detected bounding box can still satisfactorily overlap the ground-truth bounding box (region of interests). To ensure those three keypoints are detected correctly, it is required that the detected bounding box must have an Intersection-over-Union (IoU) overlap higher than 0.7 with the ground-truth bounding box, and the radius is thus determined by the size of the ground-truth bounding box. For convenience, let (m_1, n_1), (m_2, n_2) and (m_3, n_3) indicate coordinates for the top-left corner point, the centroid point and the bottom-right corner point of the detected bounding box respectively, and $(\hat{m}_1, \hat{n}_1)$, $(\hat{m}_2, \hat{n}_2)$ and $(\hat{m}_3, \hat{n}_3)$ are those of the ground-truth bounding box. The soft labels corresponding to the pixel (m, n) are defined by an unnormalized Gaussian function,

$$y_{(m,n),i} = \exp\left(-\frac{(m - \hat{m}_i)^2 + (n - \hat{n}_i)^2}{2\sigma^2} \right), \quad i \in \{1, 2, 3\} \tag{7}$$

where i is the index of a keypoint as mentioned above, and σ is $1/3$ of the radius. The soft labels $y_{(m,n),i}$ is centered at their corresponding ground-truth keypoints $(\hat{m}_i, \hat{n}_i)$ and smoothly reduces to 0 for the locations outside the radius.

Although we increase the number of positive locations by labeling the negative locations within a radius of the ground-truth location with soft labels, there remains an imbalance problem between the limited positive locations within the radius and a substantial amount of negative locations outside the radius. These easy negative locations will take over the majority of training losses and dominate the gradient. To solve this problem, we investigate the relationship between the soft labels $y_{(m,n),i}$ and the probability scores $s_{(m,n),i}$ corresponding to the location (m, n), and propose a location sensitive loss function $\mathcal{L}_{hm}$ of the keypoint heatmap,

$$\mathcal{L}_{hm} = -\sum_{i=1}^{3}\sum_{m=1}^{W}\sum_{n=1}^{H} y_{(m,n),i}(1 - s_{(m,n),i}) \log(s_{(m,n),i}) + s_{(m,n),i}(1 - y_{(m,n),i}) \log(1 - s_{(m,n),i}) \tag{8}$$

where $s_{(m,n),i}$ is the score value at location (m, n) associated with the i-th keypoint heatmap obtained in Section 3.4.

With the ground-truth keypoints $(\hat{m}_i, \hat{n}_i)$, the precision loss caused by the downsampling operations can be written as below,

$$\hat{\mathbf{O}}_i = \left(\frac{\hat{m}_i}{\alpha} - \left\lfloor\frac{\hat{m}_i}{\alpha}\right\rfloor, \frac{\hat{n}_i}{\alpha} - \left\lfloor\frac{\hat{n}_i}{\alpha}\right\rfloor\right), \quad i \in \{1, 2, 3\} \tag{9}$$

where α is the downsampling factor. Then the offset loss $\mathcal{L}_{os}$ can be formulated as,

$$\mathcal{L}_{os} = \sum_{i=1}^{3}\sum_{k=1}^{2} smooth_{L1}\left(\mathbf{O}_i^k - \hat{\mathbf{O}}_i^k\right) \tag{10}$$

where $\mathbf{O}_i$ is the offset map predicted in Section 3.4, and $smooth_{L1}$ is a robust loss function [26] defined as,

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{11}$$

Since the top-left corner point and the bottom-right corner point should be symmetric around the centroid point, the relative distance between the top-left corner point and the centroid point must be equal to that between the bottom-right corner point and the centroid point. Hence, the symmetry loss is defined as,

$$\mathcal{L}_{st} = ((m_2 - m_1) - (m_3 - m_2)) + ((n_2 - n_1) - (n_3 - n_2)) \tag{12}$$

Finally, our SATIN is trained by minimizing the combination loss $\mathcal{L}$ of the heatmap loss $\mathcal{L}_{hm}$, the offset loss $\mathcal{L}_{os}$ and the symmetry loss $\mathcal{L}_{st}$ as,

$$\mathcal{L} = \mathcal{L}_{hm} + \lambda_1\mathcal{L}_{os} + \lambda_2\mathcal{L}_{st} \tag{13}$$

where λ_1 and λ_2 are trade-off factors to balance the three loss.

4. Experiments

In this section, implementation details are introduced first. Then extensive experiments on OTB and VOT benchmark datasets are conducted to evaluate our proposed approach. Finally, ablation studies are carried out on OTB benchmark datasets to investigate how each component contributes to improve performance.

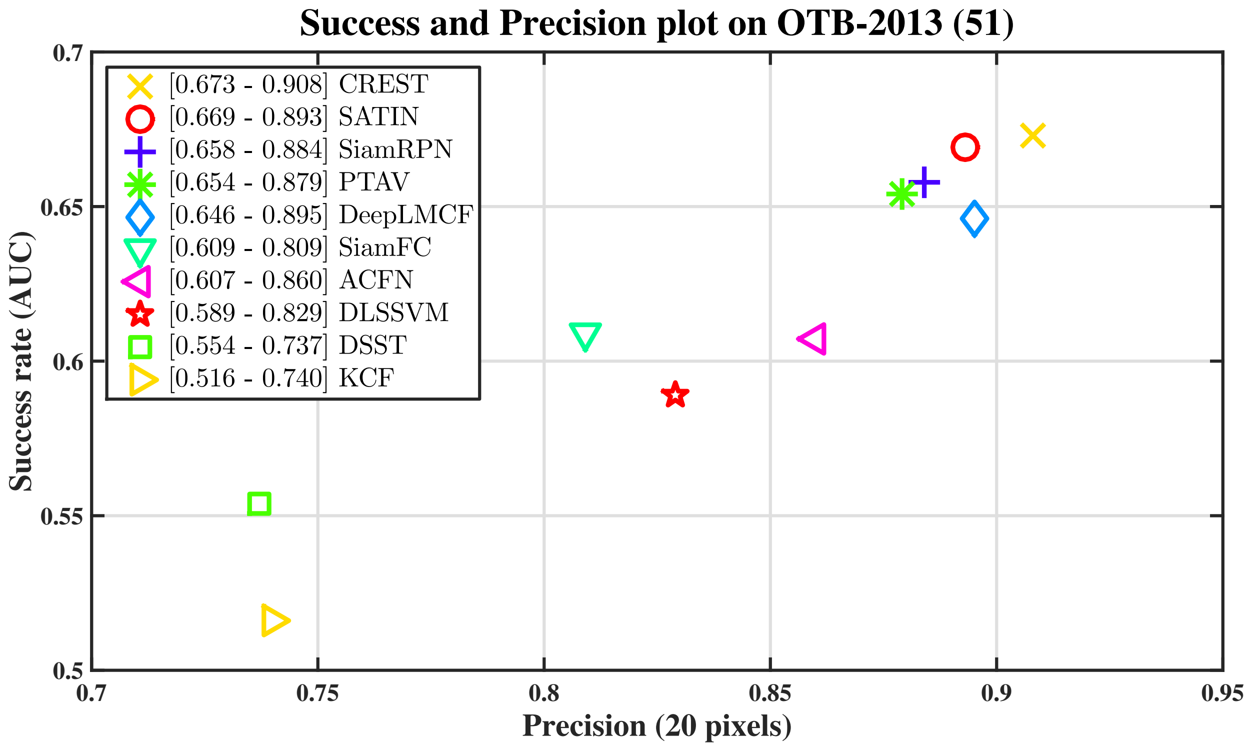

Figure 8: Success and precision performance on the OTB-2013 benchmark [35]. The first number in the legend indicates the AUC score while the second denotes DP score at error threshold of 20 pixels. The best trackers are located at the top-right corner.

4.1. Implementation details

We apply stochastic gradient descent (SGD) with the momentum of 0.9 and the weight decay of 0.005 to train SATIN offline from scratch using Image-VID [31] and YouTube-VOS [53]. Totally 50 epochs are performed with learning rate decreased in a logarithmic manner from 10^{-3} to 10^{-5}. Frame pairs are collected with interval less than 100 from the same video sequence, and exemplar images and candidate images are then cropped from frames pairs around their corresponding ground-truth bounding box with 2× and 4× padding respectively. Random translations are performed up to ±12 pixels, and the range of rescaling varies from $2^{-1/4}$ to $2^{1/4}$. If the cropped image extends exceed the size of the frame, the missing portion is filled with the mean background RGB value. After that, exemplar and candidate images are resized to the size of $127 \times 127 \times 3$ and $255 \times 255 \times 3$, respectively. The trade-off factors in Eq. 13 are set as $\lambda_1 = 1$ and $\lambda_2 = 10$. Before we train the full approach, the backbone network is pre-trained individually on ImageNet-DET [31] and COCO [54] for warm up to enhance both generalization and representation capabilities. Similar to [41]and [34], we add intermediate supervision in training without adding the intermediate predictions back to the network.

Our approach is implemented using MXNet [55], trained on an Amazon EC2 instance `p2.16xlarge` with 16 Intel Xeon E5-2686 CPU, 732GB RAM and 16 NVIDIA K80 GPU, 192GB VRAM, and tested on an Amazon EC2 instance `p2.xlarge` with an Intel Xeon E5-2686 CPU, 61GB RAM

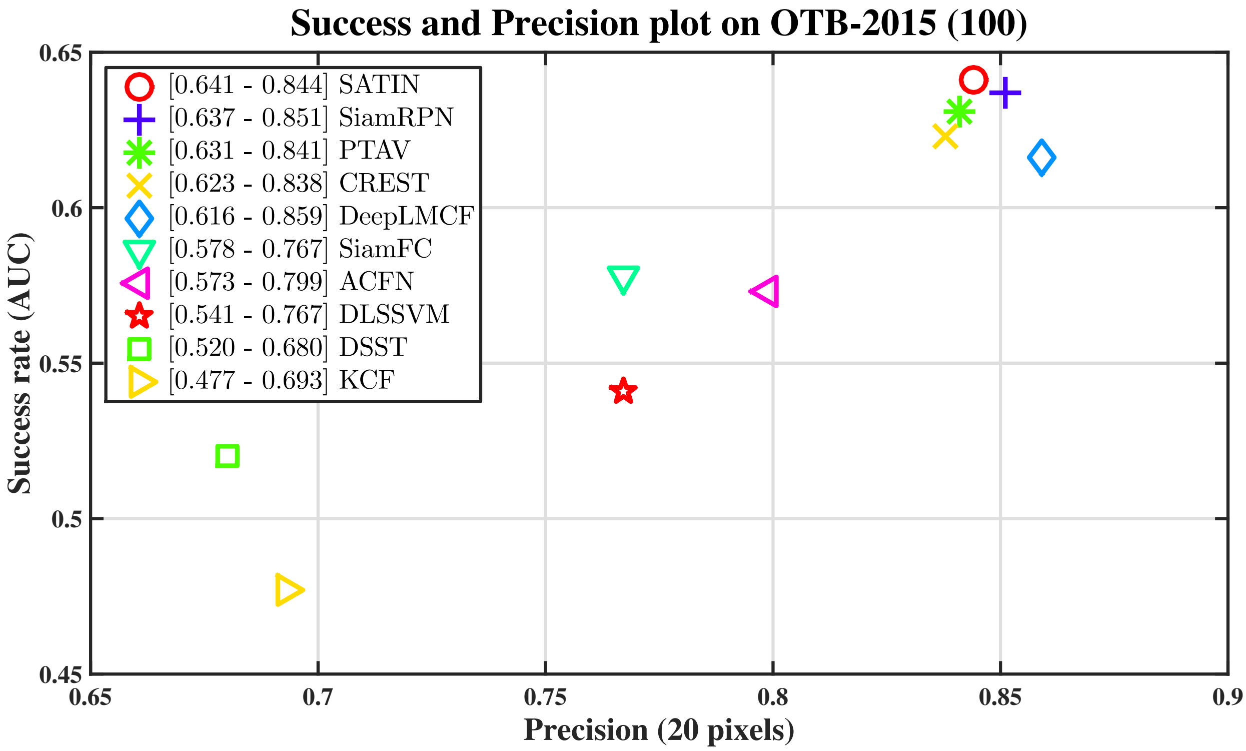

Figure 9: Success and precision performance on the OTB-2015 benchmark [36]. The first number in the legend indicates the AUC score while the second denotes DP score at error threshold of 20 pixels. The best trackers are located at the top-right corner.

and an NVIDIA K80 GPU, 12GB VRAM. All the parameters are fixed across experiments and datasets. The average tracking speed of our proposed tracker is 27 frames per second (fps) which exceeds the real-time bound of 15 fps [2] by a large margin. The code and pre-trained models will be made publicly available.

4.2. Experiments on OTB

We compare SATIN with nine state-of-the-art trackers, including ACFN [56], CREST [57], DSST [11], PTAV [58], SiamRPN [24], SiamFC [21], DeepLMCF [59], DLSSVM [60] and KCF [10], on the OTB benchmark datasets [35, 36]. Among the participants, we treat SiamFC and SiamRPN as our baseline. Two widely-used metrics: distance precision (DP) and overlap success (OS) are employed to evaluate tracking performance. DP is the percentage of frames in the video where the Euclidean distance between the detected object location and the ground-truth object location is smaller than a fixed threshold of 20 pixels. OS is the percentage of frames in the video where the overlap ratio between the detected bounding box and the ground-truth bounding box exceeds a preset threshold of 0.5. All the trackers are initialized with the ground-truth bounding box in the first frame. For better performance measure, we utilize the area-under-curve (AUC) to present the OS score within the threshold range of [0, 1] [35].

Fig. 8 illustrates the *success and precision plot* under one-pass evaluation (OPE) over all 51

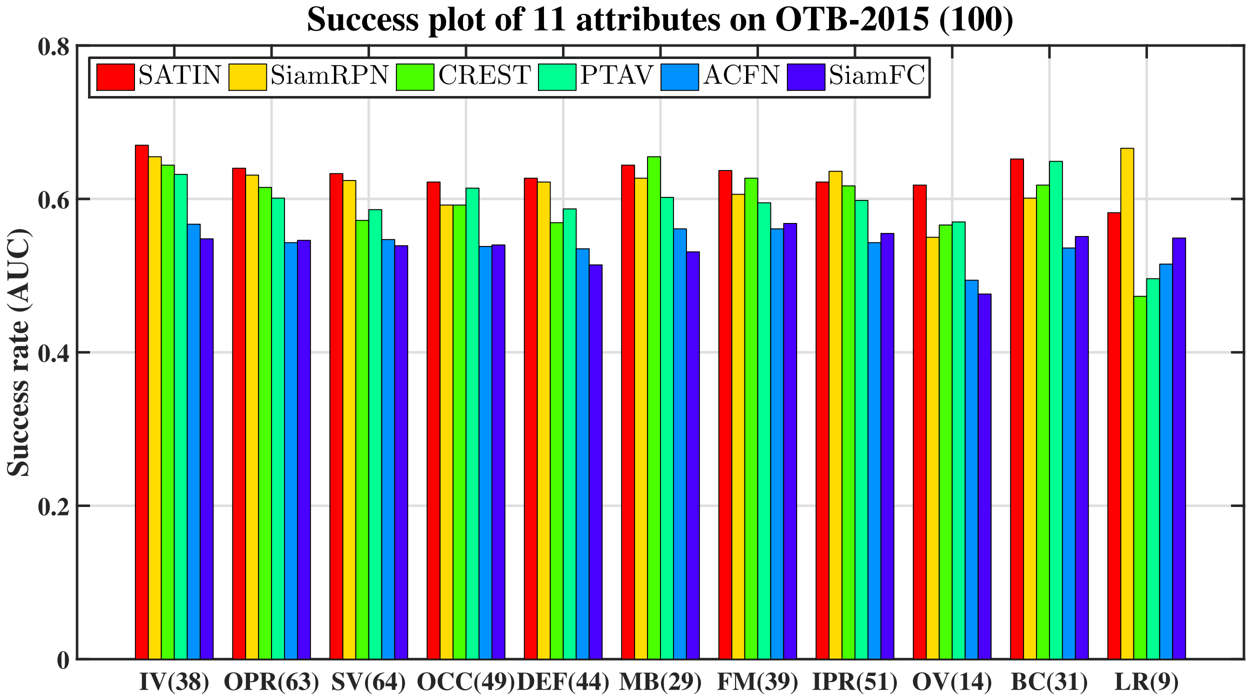

Figure 10: The success plot of 11 attributes on the OTB-2015 benchmark [36]. Best viewed in color.

fully-annotated video sequences in the OTB-2013 benchmark dataset [35]. SATIN achieves the second-best and the third-best performance in AUC and DP scores, respectively. The residual tracker CREST [57] performs favorably against all compared trackers with the best AUC score of 67.3% and the best DP score of 90.8%. KCF [10] as one of the milestones of DCF-based approaches, attains an AUC score of 51.6% and a DP score of 74.0%. DSST [11] is an extension of KCF that further increases the AUC score to 55.4%, but is not as accurate as its baseline. The groundbreaking similarity measurement tracking approach, namely SiamFC [21], gains improvements of 5.5% on AUC score and 7.2% on DP score compared to DSST. PTAV [58] improves 4.5% on AUC score and 7.0% on DP score compared to SiamFC owing to the parallel framework. ACFN provides an AUC score of 60.7% and a DP score of 86.0%. It exploits an attention mechanism to select a tracker, and is required to maintain totally 260 trackers at the same time while its performance is still left far behind SiamFC. DeepLMCF [59] and DLSSVM [60] achieve AUC scores of 64.6% and 58.9%, respectively. They both employ support vector machines (SVM) as classifiers to distinguish the target object from surrounding background. SiamRPN [24] exploits the Siamese region proposal network to get more accurate bounding boxes. It provides an AUC score of 65.8% and a DP score of 88.4%, which are slightly less accurate and robust than CREST. Compared with SiamRPN, SATIN surpasses it by absolute gains of 1.6% on AUC score and 1.1% on DP score. Overall ,the evaluation results on OTB benchmark datasets demonstrate that our proposed SATIN tracker performs well against state-of-the-art trackers over the 50 challenging video sequences.

Fig. 9 illustrates the *success and precision plot* under OPE over all the 100 fully-annotated

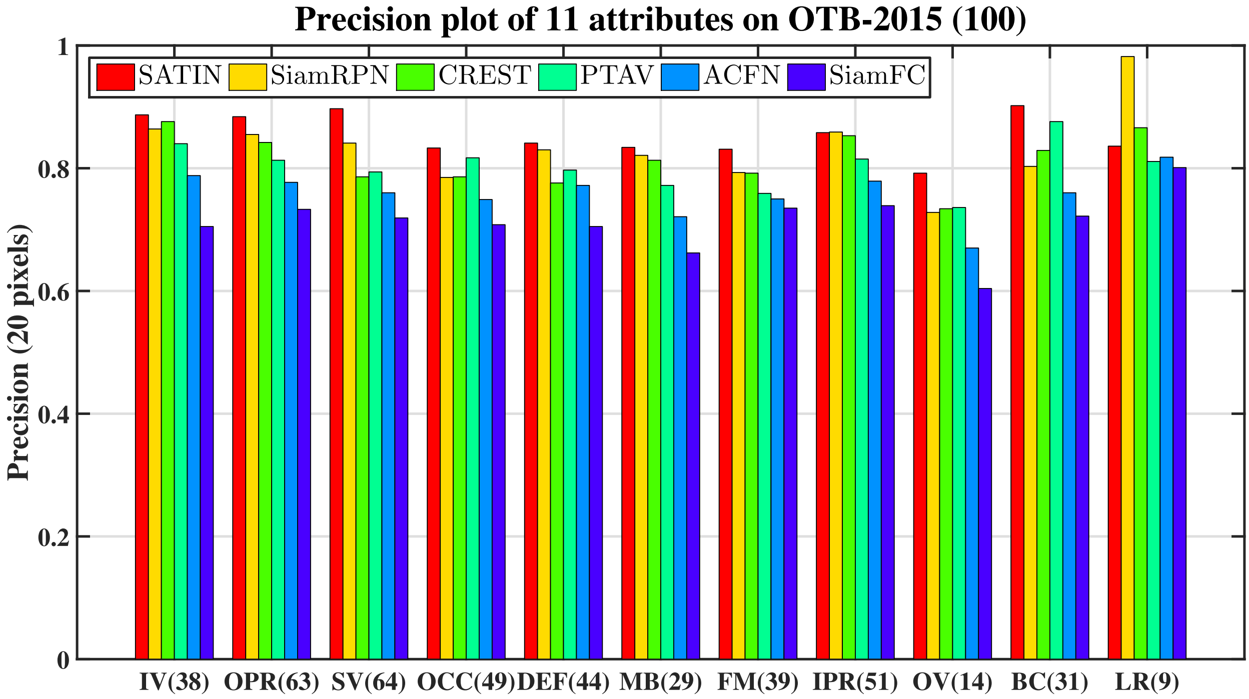

Figure 11: The precision plot of 11 attributes on the OTB-2015 benchmark [36]. Best viewed in color.

video sequences in the OTB-2015 benchmark dataset [36]. SATIN achieves the best and the third-best results in terms of AUC and DP, respectively. Compared with SiamFC, SATIN improves the AUC score from 57.8% to 64.1% and the DP score from 76.7% to 84.4%. CREST has provided the best performance on the OTB-2013 benchmark dataset, but SATIN outperforms it by amazing improvements of 1.8% and 0.6% on the AUC and DP scores after extending the dataset to 100 video sequences. In particular, SATIN is superior to the baseline SiamRPN tracker in the AUC score with an absolute gain of 1.0%. However, SiamRPN achieves a DP score of 85.1% which provides an absolute gain of 0.7% compared to SATIN. Our approach performs better than recent PTAV tracker, and gets 1.0% and 0.3% absolute gains in terms of AUC and DP scores. Although DeepLMCF obtains the best DP performance of 85.9%, its AUC score of 61.6% is much inferior to SATIN. Besides, SATIN achieves absolute AUC/DP gains of 16.4%/15.1%, 12.1%/16.4%, 10.0%/7.7% and 6.8%/4.5% over KCF, DSST, DLSSVM and ACFN, respectively. The excellent tracking performance of SATIN is primarily attributed to two aspects. First, the specially designed backbone network and the cross-attentional module are utilized to enhance the representation and generalization capabilities of SATIN. Second, SATIN tracks target objects by regressing keypoints of the bounding boxes to soft labels instead of generating redundant region proposals or designing complicated scale pyramids.

For comprehensive evaluation and analysis on the effectiveness of our approach, we compare SATIN with five state-of-the-art trackers including SiamRPN [24], CREST [57], PTAV [58], ACFN [56] and SiamFC [21] in different attributes on the OTB-2015 benchmark. Each sequence in the OTB-2015 benchmark dataset is categorized with 11 attributes. Among all the 100 fully-

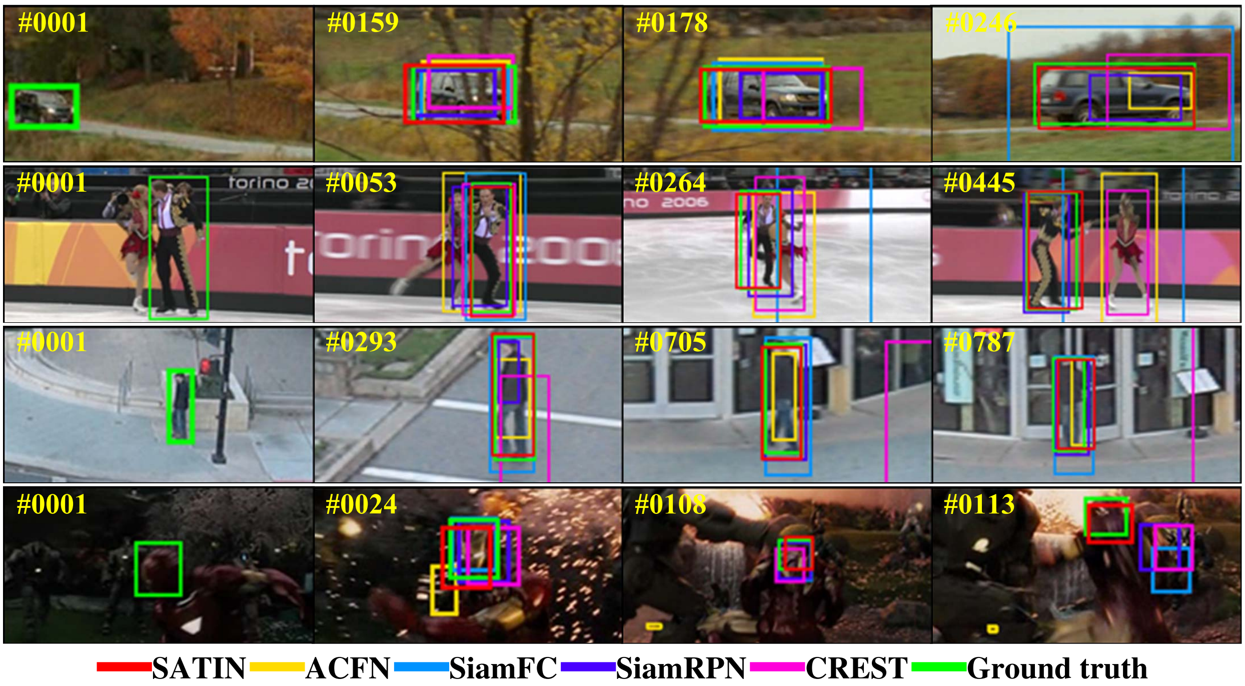

Figure 12: Example tracking results of STAIN with the comparison to state-of-the-are trackers including ACFN [56], SiamFC [21], SiamRPN [24] and CREST [57] on four challenging sequences (from top to down: *carScale*, *Skating2-2*, *Human6* and *ironman*). The ground truth is illustrated with green bounding boxes. Best viewed in color.

annotated video sequences, there are 38 sequences with illumination variation (IV), 63 sequences with out-of-plane rotation (OPR), 64 sequences with scale variation (SV), 49 sequences with occlusion (OCC), 44 sequences with deformation (DEF), 29 sequences with motion blur (MB), 39 sequences with fast motion (FM), 51 sequences with in-plane rotation (IPR), 14 sequences with out of view (OV), 31 sequences with background clutter (BC) and 9 sequences with low resolution (LR). The results of overlap success and distance precision are shown in Fig. 10 and Fig. 11, respectively. It is clear that the proposed SATIN obtains the best performance under 8 of 11 attributes. However, SATIN is slightly worse than SiamRPN on the attributes of in-plane rotation and low resolution. The main reason is that our backbone network may lose some fine-grained information through the repeated bottom-up and top-down framework. Besides, SATIN performs more accurate and robust under occlusion and background clutter with the help of attentional feature refinement, which can emphasize the informative feature regions and channels and reduce the effect of background noise to facilitate tracking tasks. Fig. 12 shows comparison results of SATIN with ACFN, SiamFC, SiamRPN and CREST on OTB benchmark datasets. SATIN performs well on all the presented video sequences including *carScale* (with attributes of SV, OCC, FM, IPR and OPR), *Skating2-2* (with attributes of SV, OCC, DEF, FM and OPR), *Human6* (with attributes of SV, OCC, DEF, FM, OPR and OV) and *ironman* (with attributes of IV, SV, OCC, MB, FM, IPR, OPR, OV, BC and LR). All the results clearly demonstrate the excellent performance of SATIN.

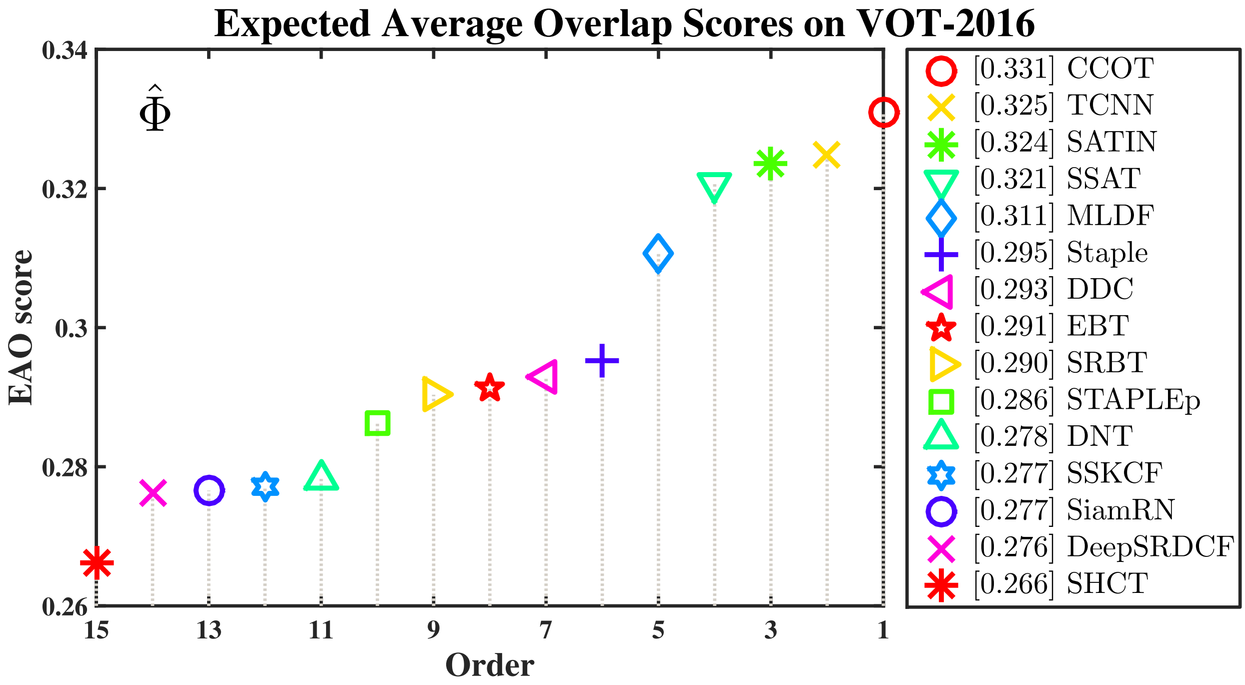

Figure 13: Expected average overlap plot on the VOT-2016 challenge dataset [37]. For presentation clarity, only the top 15 trackers with respect to the EAO score are shown in the plot. The better trackers are located at the right.

4.3. Experiments on VOT

The VOT challenge [61] is the most significant annual competition in the field of visual tracking. Each of VOT-2016 and VOT-2017 benchmark datasets [37, 38] contains 60 video sequences. We use them to evaluate our proposed approach. In the experiments, the performance is evaluated in three metrics: Accuracy, Robustness and Expected Average Overlap (EAO). Accuracy is defined as the average overlap between the detected bounding box and the ground truth bounding box during successful tracking periods. Robustness is defined as how many times tracking failures occur. EAO represents the average overlap with no re-initialization following a failure. For fair comparisons, we use the original results provided by the VOT challenge committee [1].

We compared SATIN with 70 other state-of-the-art trackers on the VOT-2016 challenge dataset, and report EAO scores of the top 15 participants in Fig. 13. Among all the trackers, CCOT [16] achieves the best EAO score of 0.331, which exploits continuous convolution operators on multi-level feature maps. It is clear that SATIN is superior to most of state-of-the-art, and obtains the third-best EAO score of 0.324, which is only 0.001 lower than the second best tracker TCNN [62]. However, there is still a gap compared with the top performer CCOT [16]. Compared with SiamFC [21] and DeepSRDCF [13], SATIN achieves absolute gains of 4.7% and 4.8% in EAO, respectively.

For the evaluation on the VOT-2017 challenge dataset, Fig. 14 reports EAO score of ours in comparison with 14 other state-of-the-art trackers. Compared with the top-ranked tracker in the

[1] `http://www.votchallenge.net/`

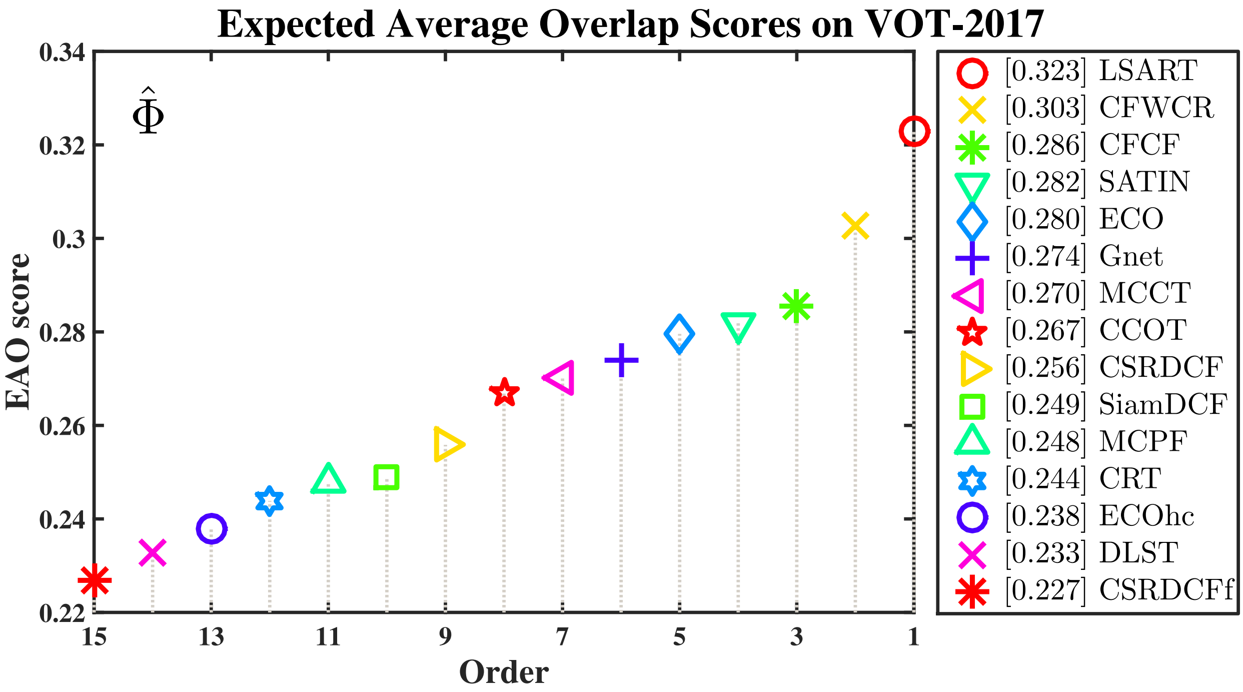

Figure 14: Expected average overlap plot on the VOT-2017 challenge dataset [38]. For presentation clarity, only the top 15 trackers with respect to the EAO score are shown in the plot. The better trackers are located at the right.

VOT-2016 challenge, namely CCOT, the proposed approach improves the EAO score from 0.267 to 0.282, obtaining an absolute gain of 1.5%. LSART [63] divides the target region into several patches, and applies a weighted combination of patch-wise similarities into a kernelized ridge regression. Although LSART provides the best EAO score of 0.323, our SATIN tracker is much simpler than it. Compared with SiamDCF which performs correlation analysis on multi-level feature maps, SATIN obtains a significant EAO gain of 3.3%. This is mainly attributed to our tracker takes advantages of the benefits of both global and local contextual representations across multiple scales.

Table 1 reports the detailed results of SATIN against CFCF [23], ECO [15], CCOT [16], CSRDCF [48], SiamFC [21], Staple [64], KCF [10] and DSST [11] with respect to Accuracy, Robustness and EAO score. Among the nine compared trackers, CFCF obtains the best accuracy score of 0.509 and ECO provides the best robustness score of 1.117. SATIN achieves the third-best accuracy score of 0.491 and the fifth-best robustness score of 1.344, which are comparable to CCOT and CSRDCF, respectively. Compared with SiamFC which has an EAO score of 0.188, SATIN substantially obtains a substantial gain of 9.4% in EAO and demonstrates its superiority in terms of accuracy and robustness. In addition, compared to the most robust tracker, i.e. ECO, SATIN achieves a significant improvement in accuracy with an absolute gain of 0.8%. All the above experiments show that our proposed approach performs well compared with other state-of-the-art trackers.

Table 1: Comparisons between STAIN and eight state-of-the-art trackers on the VOT-2017 challenge dataset. The results are presented in terms of expected average overlap (EAO), accuracy and robustness. The first, second and third best values are highlighted in **red**, **blue** and **green**, respectively.

Tracker	EAO	Accuracy	Robustness
CFCF [23]	**0.286**	**0.509**	**1.169**
ECO [15]	**0.281**	0.483	**1.117**
CCOT [16]	0.267	**0.493**	1.315
CSRDCF [48]	0.256	0.488	**1.309**
SiamFC [21]	0.188	0.502	2.049
Staple[64]	0.169	0.530	2.506
KCF [10]	0.135	0.447	2.577
DSST[11]	0.079	0.395	5.429
SATIN	**0.282**	**0.491**	1.344

4.4. Ablation studies

To validate our tracker and analyze the effectiveness of each component, several ablative implementations of SATIN as well as the baseline tracker SiamFC [21] are evaluated on OTB benchmarks. The detailed evaluation results are illustrated in Table 2.

Our full tracking approach (SATIN) outperforms all other implementations. The effectiveness of lightweight hourglass networks is evaluated by comparison with the variant SATIN$_{VGG}$ which exploits VGG-M [29] as the backbone network. Compared with the full SATIN tracker, the performance of SATIN$_{VGG}$ drops more than 2.6% and 2.9% in terms of AUC and DP, which proves that our backbone network can learn more contextual representations than the off-the-shelf networks. To investigate how the cross-attentional module contributes to SATIN, we implement three alternative variants, i.e., SATIN$_{NAA}$ (SATIN without all attention), SATIN$_{NSA}$ (SATIN without the spatial attention) and SATIN$_{NCA}$ (SATIN without the channel-wise attention). In addition, we also implement a variant SATIN$_{SENet}$ that incorporates the recent SENet [32] to learn channel-wise attention. SATIN$_{NSA}$ obtains absolute AUC and DP gains of 1.5% and 1.0% compared with SATIN$_{SENet}$. This is mainly caused by different pooling strategies of these two variants. Compared with SATIN$_{NAA}$, after integrating both channel-wise and spatial attentional information in the full approach, both robustness and accuracy are dramatically improved, with gains of almost 3.3% and 3.2% in AUC and DP scores, respectively. Without corner pooling operation (SATIN$_{NCP}$), inaccurate corner point detections significantly deteriorate the tracking results. This fact demonstrates that the corner pooling operations is more critical for detecting corner points. Specifically, we observe that the AUC scores of SATIN$_{NCP}$ is inferior to that of the baseline SiamFC with 1.2% drop. But the DP score is around 2.4% higher than that of SiamFC owing to that centroid point detection does not need corner pooling operation. These results indicate that each component brings

Table 2: Ablation studies of several variations of our tracker on OTB benchmarks using AUC and Precision metrics. The first, second and third best values are highlighted in **red**, **blue** and **green**, respectively.

Trackers	OTB-2015		OTB-2013	
	AUC (%)	Precision (%)	AUC (%)	Precision (%)
SiamFC [21]	57.8	76.7	60.9	80.9
SATIN$_{VGG}$	**61.9**	**82.7**	**64.3**	86.4
SATIN$_{NCP}$	57.2	79.1	59.7	82.5
SATIN$_{NAA}$	60.8	81.2	62.4	84.3
SATIN$_{SENet}$	61.1	82.4	63.2	**87.1**
SATIN$_{NSA}$	**62.6**	**83.4**	**64.5**	**87.6**
SATIN$_{NCA}$	61.5	82.2	63.7	86.5
SATIN	**64.1**	**84.4**	**66.9**	**89.3**

individual improvement, and all of them work together to achieve surprisingly excellent tracking performance.

5. Conclusions

In this paper, we propose a new deep architecture named SATIN for high performance visual tracking. It utilizes Siamese lightweight hourglass networks as the backbone network, which can extract more global and local contextual representations at multiple scales in a repeated bottom-up and top-down manner. We consider generic visual tracking as a set of keypoint detection tasks, i.e., SATIN tracks any target object by detecting three keypoints of its bounding box, including the top-left corner point, the centroid point and the bottom-right corner point. Thus, our approach eliminates the need for designing anchor boxes or multi-scale pyramids. Meanwhile, a cross-attentional module is employed to heuristically learn what and where to emphasize or suppress along channel-wise and spatial dimensions on object representations, therefore promote the tracking performance further. Without bells and whistles, our SATIN tracker achieves state-of-the-art results on four popular benchmark datasets. Furthermore, SATIN runs at 27 fps, which is far above real-time frame rate.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 31701187, the Science and Technology Planning Program of Guangdong Province under Grant No. 2016B090918047, and Promotional Credit from Amazon Web Service, Inc.

References

[1] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, ACM Computing Surveys 38 (4) (2006) 13.

[2] H. Yang, L. Shao, F. Zheng, L. Wang, Z. Song, Recent advances and trends in visual tracking: A review, Neurocomputing 74 (18) (2011) 3823–3831.

[3] 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 (7) (2014) 1442–1468.

[4] P. Li, D. Wang, L. Wang, H. Lu, Deep visual tracking: Review and experimental comparison, Pattern Recognition 76 (2018) 323–338.

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 2544–2550.

[6] J. Henriques, R. Caseiro, P. Martins, J. Batista, Exploiting the circulant structure of tracking-by-detection with kernels, in: European conference on computer vision (ECCV), Springer, 2012, pp. 702–715.

[7] Z. Zhu, W. Wu, W. Zou, J. Yan, End-to-end flow correlation tracking with spatial-temporal attention, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[8] P. Gao, Y. Ma, K. Song, C. Li, F. Wang, L. Xiao, Large margin structured convolution operator for thermal infrared object tracking, in: International Conference on Pattern Recognition (ICPR), 2018, pp. 2380–2385.

[9] M. Danelljan, F. S. Khan, M. Felsberg, J. van de Weijer, Adaptive color attributes for real-time visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2014, pp. 1090–1097.

[10] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596.

[11] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Discriminative scale space tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (8) (2017) 1561–1575.

[12] Y. Li, J. Zhu, A scale adaptive kernel correlation filter tracker with feature integration., in: European Conference on Computer Vision (ECCV), Springer, 2014, pp. 254–265.

[13] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Learning spatially regularized correlation filters for visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, pp. 4310–4318.

[14] H. Kiani Galoogahi, A. Fagg, S. Lucey, Learning background-aware correlation filters for visual tracking, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 1135–1143.

[15] M. Danelljan, G. Bhat, S. F. Khan, M. Felsberg, Eco: Efficient convolution operators for tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[16] M. Danelljan, A. Robinson, F. S. Khan, M. Felsberg, Beyond correlation filters: Learning continuous convolution operators for visual tracking, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 472–488.

[17] P. Gao, Y. Ma, C. Li, K. Song, Y. Zhang, F. Wang, L. Xiao, Adaptive object tracking with complementary models, IEICE Transactions on Information and Systems E101-D (11) (2018) 2849–2854.

[18] Y. Ma, C. Yuan, P. Gao, F. Wang, Efficient multi-level correlating for visual tracking, in: Asian Conference on Computer Vision (ACCV), Springer, 2018.

[19] P. Gao, Y. Ma, C. Li, K. Song, F. Wang, L. Xiao, A complementary tracking model with multiple features, Proceedings of SPIE 10836 (18) (2018) 1–5.

[20] R. Tao, E. Gavves, A. W. M. Smeulders, Siamese instance search for tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 1420–1429.

[21] L. Bertinetto, J. Valmadre, J. Henriques, A. Vedaldi, P. H. S. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 850–865.

[22] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P. H. S. Torr, End-to-end representation learning for correlation filter based tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 2805–2813.

[23] E. Gundogdu, A. A. Alatan, Good features to correlate for visual tracking, IEEE Transactions on Image Processing 27 (5) (2018) 2526–2540.

[24] B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 8971–8980.

[25] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: European Conference on Computer Vision (ECCV), Springer, 2018, pp. 103–119.

[26] R. Girshick, Fast r-cnn, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[27] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Annual Conference on Neural Information Processing Systems (NIPS), 2015, pp. 91–99.

[28] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Annual Conference on Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556v6.

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 770–778.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, F.-F. Li, Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

[32] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018.

[33] S. Woo, J. Park, J.-Y. Lee, I. So Kweon, Cbam: Convolutional block attention module, in: European Conference on Computer Vision (ECCV), Springer, 2018, pp. 3–19.

[34] H. Law, J. Deng, Cornernet: Detecting objects as paired keypoints, in: European Conference on Computer Vision (ECCV), 2018, pp. 734–750.

[35] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2013, pp. 2411–2418.

[36] Y. Wu, J. Lim, M.-H. Yang, Object tracking benchmark, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (9) (2015) 1834–1848.

[37] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, The visual object tracking vot2016 challenge results, in: European Conference on Computer Vision (ECCV), Springer, 2016.

[38] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojíř, et al., The visual object tracking vot2017 challenge results, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 1949–1972.

[39] P. Gao, Y. Ma, K. Song, C. Li, F. Wang, L. Xiao, Y. Zhang, High performance visual tracking with circular and structural operators, Knowledge-Based Systems 161 (2018) 240–253.

[40] C. Ma, J.-B. Huang, X. Yang, M.-H. Yang, Hierarchital convolutional features for visual tracking, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2015.

[41] A. Newell, K. Yang, J. Deng, Stacked hourglass networks for human pose estimation, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 483–499.

[42] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2758–2766.

[43] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861.

[44] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50× fewer parameters and < 0.5 mb model size, arXiv preprint arXiv:1602.07360.

[45] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1251–1258.

[46] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 4293–4302.

[47] H. Li, P. Xiong, J. An, L. Wang, Pyramid attention network for semantic segmentation, in: British Machine Vision Conference (BMVC), Springer, 2018.

[48] A. Lukežič, T. Vojíř, L. Čehovin, J. Matas, M. Kristan, Discriminative correlation filter with channel and spatial reliability, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 4847–4856.

[49] C. Sun, D. Wang, H. Lu, M.-H. Yang, Learning spatial-aware regressions for visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 8962–8970.

[50] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, S. Maybank, Learning attentions: residual attentional siamese network for high performance online visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4854–4863.

[51] R. Liu, D. Wang, Y. Han, X. Fan, Z. Luo, Adaptive low-rank subspace learning with online optimization for robust visual tracking, Neural Networks 88 (2017) 90–104.

[52] N. Wang, J. Shi, D.-Y. Yeung, J. Jia, Understanding and diagnosing visual tracking systems, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, pp. 3101–3109.

[53] E. Real, J. Shlens, S. Mazzocchi, X. Pan, V. Vanhoucke, Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 5296–5305.

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision (ECCV), Springer, 2014, pp. 740–755.

[55] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, arXiv preprint arXiv:1512.01274.

[56] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, J. Y. Choi, Attentional correlation filter network for adaptive visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[57] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. Lau, M.-H. Yang, Crest: Convolutional residual learning for visual tracking, in: IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2574–2583.

[58] H. Fan, H. Ling, Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking, in: IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5486–5494.

[59] M. Wang, Y. Liu, Z. Huang, Large margin object tracking with circulant feature maps, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[60] J. Ning, J. Yang, S. Jiang, L. Zhang, M.-H. Yang, Object tracking via dual linear structured svm and explicit feature map, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 4266–4274.

[61] M. Kristan, J. Matas, A. Leonardis, T. Vojíř, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, L. Čehovin, A novel performance evaluation methodology for single-target trackers, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (11) (2016) 2137–2155.

[62] H. Nam, M. Baek, B. Han, Modeling and propagating cnns in a tree structure for visual tracking, arXiv preprint arXiv:1608.07242.

[63] C. Sun, H. Lu, M.-H. Yang, Learning spatial-aware regressions for visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[64] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, P. H. Torr, Staple: Complementary learners for real-time tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 1401–1409.