This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Correlation-Guided Attention for Corner Detection Based Visual Tracking

Fei Du Peng Liu Wei Zhao * Xianglong Tang
School of Computer Science and Technology, Harbin Institute of Technology, China
{feiaxyt, pengliu, zhaowei, tangxl}@hit.edu.cn

Abstract

Accurate bounding box estimation has recently attracted much attention in the tracking community because traditional multi-scale search strategies cannot estimate tight bounding boxes in many challenging scenarios involving changes to the target. A tracker capable of detecting target corners can flexibly adapt to such changes, but existing corner detection based tracking methods have not achieved adequate success. We analyze the reasons for their failure and propose a state-of-the-art tracker that performs correlation-guided attentional corner detection in two stages. First, a region of interest (RoI) is obtained by employing an efficient Siamese network to distinguish the target from the background. Second, a pixel-wise correlation-guided spatial attention module and a channel-wise correlation-guided channel attention module exploit the relationship between the target template and the RoI to highlight corner regions and enhance features of the RoI for corner detection. The correlation-guided attention modules improve the accuracy of corner detection, thus enabling accurate bounding box estimation. When trained on large-scale datasets using a novel RoI augmentation strategy, the performance of the proposed tracker, running at a high speed of 70 FPS, is comparable with that of state-of-the-art trackers in meeting five challenging performance benchmarks.

1. Introduction

Visual object tracking is one of the fundamental problems in computer vision, and it has attracted increasing attention in recent years. The goal of object tracking is to locate a target at any point in a video based on the state of the target in the first frame. The ability to do this online would have multiple applications in human-computer interactions, video surveillance, and unmanned control systems. Although significant progress has been made, factors such as occlusion, deformation, and scale variation still pose formidable challenges.

*Corresponding author

A tracker is normally required both to locate the target robustly and to estimate its state (frequently represented by a bounding box) accurately. Most trackers focus on target location through the construction of robust classifiers, ignoring target-state estimation. As a result, most trackers cannot handle severe target deformation, because they rely on a multi-scale search strategy [25, 7] to obtain the bounding box. Estimating the bounding box would solve this problem, and a number of ways to do this have been proposed, including the linear regression model [32], the region proposal network (RPN) [24], and the intersection over union (IoU) overlap prediction network [6]. As a bounding box is determined by its top-left and bottom-right corners, one straightforward way to estimate it is to predict the coordinates of these two points. GOTURN [15] directly estimates the corner coordinates using a fully connected network. SATIN [12] employs a corner pooling strategy [22] in a Siamese network [1] to predict heatmaps and location offsets for target corners. By detecting corners, the tracker can flexibly adapt to deformations and scale changes in the target. However, approaches based on corner detection have not yet achieved a level of performance comparable to that of the other state-of-the-art trackers [46, 23, 6].

Existing corner detection-based tracking methods have several limitations. First, methods that directly detect corners are complicated by the need to resolve ambiguities: objects in the background also have corners, and the tracker may have difficulty distinguishing them from the corners of the target being tracked. Second, existing methods do not effectively explore the relationship between the template and the test image. As the target is unknown beforehand, template appearance information is often integrated into the test image representations through an integration component. In recent Siamese trackers [1, 37], a cross-correlation operation is used to calculate similarity between the template and the test image. The target center is located by finding the maximum response. However, spatial information about the corners is not explicitly encoded in the correlation results. Third, existing methods do not benefit from the powerful representation capacities of deeper networks (like ResNet [14]), using either a shallow backbone network [15]

or a light-weight hourglass network [12] instead.

In view of these limitations, we propose to learn correlation-guided attention, including pixel-wise correlation-guided spatial attention and channel-wise correlation-guided channel attention, in a two-stage corner detection network for accurate visual tracking. In the first stage, the target is distinguished from the background by using a light-weight Siamese tracking module. An RoI that is expected to contain the target is constructed at this stage. In the second stage, the bounding box is accurately estimated by detecting target corners in the constructed RoI. Such two-stage detection significantly mitigates the ambiguous corner localization problem.

The key innovation of this work is exploiting the relationship between the template and the RoI for improved accuracy in corner detection. To achieve this, we propose using a pixel-wise correlation-guided spatial attention module and a channel-wise correlation-guided channel attention module. Instead of the traditional cross-correlation, we employ pixel-wise correlation to calculate the similarity between each pixel of the template feature maps and all pixels of the RoI feature maps, thereby encoding spatial information about the corners in pixel-wise similarity maps. A spatial attention module is employed to learn spatial maps that highlight regions corresponding to the corners by taking the pixel-wise similarity maps as input. As discussed in [23], some channels have a high response to targets in a specific category, while other channels have negligible responses. We employ channel-wise correlation to calculate the channel-wise similarity maps, which encode the importance information of the various channels. A channel attention module is then employed to learn a channel descriptor that emphasizes informative channels and suppresses the useless ones by taking the channel-wise similarity maps as input. Through attentional learning, the relationship of the template and the RoI can be better exploited to help in locating the corners and enhancing the features of the RoI. In addition, target-specific information is incorporated via the proposed attention module, further helping to distinguish the target from the background in the RoI.

During training, the Siamese network generates several RoIs in all defined positive positions to train the attention module and corner-detection network extensively. To boost accuracy, we add some RoI augmentation during training. This is similar to the data augmentation used in training samples; however, our augmentation is conducted on RoIs. Our results show that RoI augmentation further increases the accuracy of bounding box estimation.

Using ResNet-50 [14] as the backbone, we train our tracker end-to-end on large-scale datasets. Extensive experiments on five large tracking benchmarks, including OTB2015 [43], VOT2018 [21], UAV123 [30], LaSOT [10], and TrackingNet [31], show that the performance of our tracker is comparable with that of the recent state-of-the-art trackers, while running at a high speed of 70 FPS.

2. Related work

We provide a brief review of recent developments in visual tracking, and discuss bounding box estimation strategies and attention mechanisms used in this field.

2.1. Visual tracking

There are two main kinds of tracker in current use: correlation filter (CF)-based trackers [4, 16, 9, 5] and Siamese network-based trackers [1, 37, 24]. CF-based trackers learn the correlation filters by solving a ridged regression problem. Many other techniques, such as multi-scale search [25, 7], boundary effects mitigation [8], multiple feature integration [9, 5], and multi-kernel training [36], have been proposed to improve the performance of CF-based trackers. However, these complex models and deep features have an adverse impact on efficiency, and practical applications often require the tracker to be highly efficient.

Siamese network based methods have recently attracted much attention in the tracking community. A Siamese tracker learns a similarity matching function by performing cross-correlation between feature representations learned from template and test images [1]. These trackers are usually very efficient, since they utilize the target in the first frame as the template and do not perform online updates. The Siamese architecture is widely used in the literature[37, 38, 13, 24, 46].

2.2. Bounding box estimation strategy

Tracking results are often represented by a rectangular bounding box. Most trackers fix the aspect ratio of the target, and resort to multi-scale search to estimate its state. This strategy is feasible in situations where targets only undergo scale changes. However, the pose and viewpoint of the object may also change with time. It is therefore desirable that trackers should be able to estimate a tight bounding box for the target. SiamRPN [24] and ATOM [6], using respectively an offline trained bounding box regression network and Intersection over Union (IoU) overlap maximization architecture, have both shown the ability to estimate accurate bounding boxes. GOTURN [15] and SATIN [12] detect corners using, respectively, a fully connected network and the cross-correlation operation, but these straightforward methods of tracking via corner detection have not performed competitively. We analyze the reasons for their inferior performance, and develop a high performance corner detection-based tracking method by learning correlation-guided attention in a two-stage corner detection network.

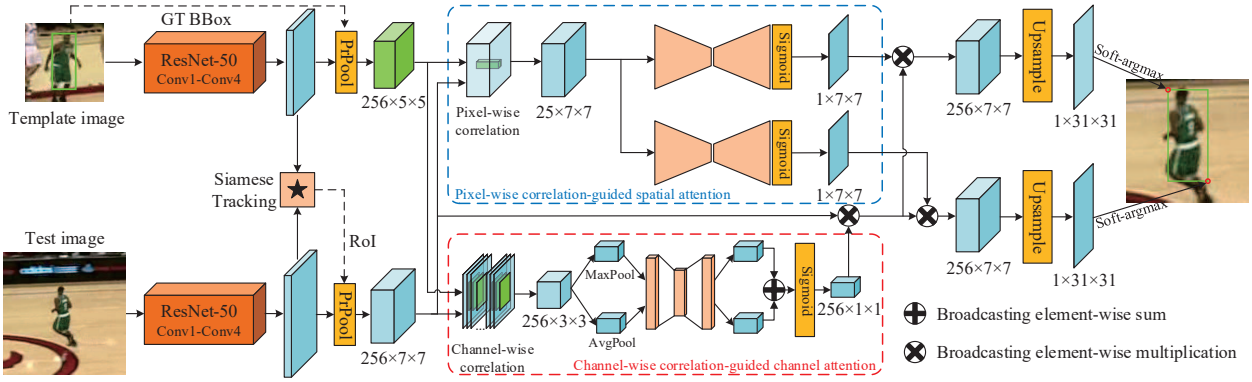

Figure 1. Framework of the proposed method. An RoI is first constructed from the result of a Siamese tracking module. Features of the template and the RoI are then extracted by two PrPool layers. The pixel-wise correlation-guided spatial attention module and channel-wise correlation-guided channel attention module are exploited to highlight corner regions and obtain enhanced features. Feature maps are then upsampled to obtain heatmaps for the two corners. Finally, two soft-argmax functions are employed to calculate corner coordinates.

2.3. Attentional mechanism

Attentional mechanisms are frequently suggested as a means of improving classification-task performance. Hu *et al*. [17] proposed a Squeeze-and Excitation module to learn channel-wise attention. Woo *et al*. [42] proposed learning spatial and channel attention simultaneously in a convolutional block attention module. In the specific field of visual tracking, CSR-DCF [28] constructs a foreground spatial map to constrain correlation filter learning and calculates the channel reliability values of weighted sum correlation response maps. RASNet [39] proposed learning spatial and channel-wise attention in a Siamese network. The purpose of attentional learning in visual tracking is usually to emphasize especially informative channels or spatial regions, so features are usually weighted in a channel-wise or a spatial manner. Most attention modules are self-contained, meaning that attention is learned only from the features to be weighted. In contrast, we learn attention from the correlation result of the template and the RoI, and use this learned attention to enhance features of the RoI. This strategy effectively exploits the relationship between the template and the RoI to highlight corner regions and to improve the discriminative power of features.

3. Proposed method

The framework of the proposed tracker is illustrated in figure 1. We take inspiration from the Grid R-CNN [27], which utilizes a top-down two-stage detector for object detection, first obtaining several RoIs though an RPN module and then extracting features from each RoI to predict grid points. As we only track one target, we use a Siamese tracking module to estimate one RoI expected to contain the target. We then employ a corner detection module to predict the top-left and bottom-right corners of this RoI. By this procedure, the target is distinguished from the background, so corner detection is less affected by background distractors. For detecting corners, we propose a two-fold correlation-guided attention module, consisting of a pixel-wise correlation-guided spatial attention module and a channel-wise correlation-guided channel attention module. The two-fold module is able to integrate target-specific information and enhance features by highlighting corner regions and informative channels, significantly improving the accuracy of corner detection. The coordinates of each corner are obtained by applying a soft-argmax function [29] on an estimated heatmap. The Siamese tracking module and the corner detection module share the same backbone network (ResNet-50) and are trained end-to-end on large-scale datasets.

3.1. Siamese tracking

In our two-stage tracker, the target is first distinguished from the background with the help of a Siamese tracking module. This module employs a light-weight Siamese architecture to estimate the initial state of the target. The target state is then utilized to construct the RoI for corner detection. Without the tracking module, the tracker would have to search the whole image for the target, and could easily drift away from the correct location because of the ambiguous corner localization problem. One simple way to construct the RoI is to locate the target using the SiamFC [1] network and draw an RoI based on its previously estimated size, but this will lead to accumulating errors if the estimate is not accurate. Thus, we also perform bounding box estimation in the first stage. The bounding box regression branch is similar to that of [20], in which the distance offset is directly mapped to the ground truth box with the coordinates of the offset value. The detailed architecture of the Siamese tracking module is shown in the supplementary material. Given a ground truth bounding box $G = (x_1, y_1, x_2, y_2)$, the expected network outputs of an

offset-map cell (x_c, y_c) are:

$$\begin{aligned} t_{x_1} &= (s \cdot x_c - x_1)/k, \quad t_{y_1} = (s \cdot y_c - y_1)/k, \\ t_{x_2} &= (x_2 - s \cdot x_c)/k, \quad t_{y_2} = (y_2 - s \cdot y_c)/k, \end{aligned} \tag{1}$$

where s denotes the stride of the network, $k = s \cdot s$ is a normalization factor, and t_{x_1}, t_{y_1}, t_{x_2}, t_{y_2} respectively denote the expected outputs of the cell (x_c, y_c) in the four offset maps. This function maps the coordinates of the cell to the test image, then finds the normalized distance offset between G and the projected coordinates. During training, the widely used smooth L_1 loss function [24] is chosen to calculate the loss L_{reg} between the expected and true network outputs. Note that only cells in the positive area contribute to the calculation of the regression loss. Denoting the projection of the ground truth bounding box on the distance offset maps as $G' = (x_1/s, y_1/s, x_2/s, y_2/s)$, the positive area $R^{pos} = (x_1^p, y_1^p, x_2^p, y_2^p)$ within G' is given by:

$$\begin{aligned} x_1^p &= c_x - 0.5\alpha_p w_t, \quad y_1^p = c_y - 0.5\alpha_p h_t, \\ x_2^p &= c_x + 0.5\alpha_p w_t, \quad y_2^p = c_y + 0.5\alpha_p h_t, \end{aligned} \tag{2}$$

where (c_x, c_y) is the center of G', w_t and h_t are respectively the width and height of G', and α_p is the shrunk factor. To train the similarity learning branch, we use another shrunk factor α_n to define a negative area R^{neg}. Cells on the similarity map that are outside the negative area are defined as negative cells; any cells that are not defined are ignored during training. With both positive and negative cells, we use the conventional logistic loss L_{sim} [1] to train the similarity learning branch.

We do not have feature pyramid representations as in [20], so the bounding box estimation is not very accurate (see section 4.1). In spite of this, it is effective at distinguishing the target from surrounding distractors. Although a more sophisticated tracking model can be used, we employ this light-weight model for efficient tracking. Because the estimated bounding box may not cover the entire target, we construct an RoI by enlarging the box. Let w_r and h_r denote the width and height of the estimated box; we construct a square RoI with a side of length $l = \sqrt{(w_r + t_r) \times (h_r + t_r)}$, where $t_r = (w_r + h_r)/2$.

3.2. Correlation-guided attention for corner detection

The bounding box of the target is obtained by predicting the target corners of the initially estimated RoI. The architecture of the correlation-guided attentional corner detection module is illustrated in figure 1. The network has two branches: the template branch and the test branch. The test branch takes as input the test image x, which contains the target we want to detect, and extracts general appearance representations through a ResNet-50 backbone network. This is followed by a PrPool (Precise RoI Pooling [19]) layer with the estimated RoI bounding box to extract feature $\phi_g(x)$ for this RoI. In object detection, $\phi_g(x)$ is directly used to predict target corners, as in [27]. However, this is infeasible in generic visual tracking, because without the supervision of the target template we lack information about the nature of the tracked object. In some Siamese trackers, concatenation and cross-correlation of features from the two branches are employed to integrate the template with the test image. However, our experiments show that these naive integration approaches for corner detection are suboptimal (see section 4.1). We attribute this suboptimal performance to the lack of effective features that not only have discriminative power but also encode spatial information about corners. Thus, we argue, better exploitation of the relationship between the template and the test image is the key to corner detection-based tracking. In this paper, we propose a novel correlation-guided attention architecture to achieve this. Before involving the test image, however, the template appearance representations are extracted from a template image z in the template branch. The architecture (ResNet-50 backbone followed by a PrPool layer) that has already been described is employed to extract generic template features $\phi_g(z)$.

Channel-wise correlation-guided channel attention: A convolutional feature channel is often activated by a certain type of visual pattern. Some channels are of little importance when learning semantic features. Channel attention can be viewed as a tool to recalibrate channel-wise feature responses [17], making the features more discriminative. We first adopt a channel-wise correlation operation to learn the similarity between the template and different regions of the RoI in a channel-wise manner, and then learn channel attention from the similarity maps. If one channel of the similarity maps has a high response, it is useful in distinguishing the tracked target and should receive a higher weight. Given $\phi_g(x) \in \mathbb{R}^{C \times H_x \times W_x}$ and $\phi_g(z) \in \mathbb{R}^{C \times H_z \times W_z}$, the result of channel-wise correlation is

$$f_c(z, x) = \phi_g(z) \star_c \phi_g(x), \tag{3}$$

where $\star_c$ denotes channel-wise correlation and $f_c \in \mathbb{R}^{C \times (H_x - H_z + 1) \times (W_x - W_z + 1)}$ is the correlation result. After calculating $f_c(z, x)$, we squeeze the spatial dimension of the result through a global max-pooling layer and a global average-pooling layer, generating two different channel descriptors: the max-pooling feature f_c^{max} and the average-pooling feature f_c^{avg}. As suggested in [42], max-pooling and average-pooling gather different clues about the channel importance. The pooled features are further fed into a shared multi-layer perceptron (MLP) to capture channel-wise dependencies, increasing the effectiveness of the resulting channel descriptors. The learned descriptors are merged using element-wise summation, and the final output of the channel-wise attention is obtained by normalizing

the output range to [0, 1] with a sigmoid activation function. The learning process can be expressed by

$$A_c = \sigma(MLP(f_c^{max}) + MLP(f_c^{avg})), \tag{4}$$

where σ denotes the sigmoid function and $A_c \in \mathbb{R}^{C\times 1\times 1}$ is the channel-wise attention.

Pixel-wise correlation-guided spatial attention: Spatial attention is learned to focus attention on informative spatial regions. For corner detection, the network should focus on the top-left and bottom-right regions of the target. However, it is not easy to learn such an attentional map just by back-propagation using the detection loss. In our work, we adopt a pixel-wise correlation between the template and the RoI to first calculate pixel-wise similarity. Before adopting the correlation, we reshape the template feature $\phi_g(z)$ to size $(H_z W_z) \times C \times 1 \times 1$. The template consists of $H_z \times W_z$ pixels and the feature for each pixel has size $C \times 1 \times 1$. Each feature is correlated with the feature maps $\phi_g(x)$ of the test image, resulting in $H_z \times W_z$ similarity maps with size $H_x \times W_x$. The pixel-wise correlation is given by

$$f_p(z, x) = \phi_g(z) \star_p \phi_g(x), \tag{5}$$

where $\star_p$ denotes the pixel-wise correlation operation and $f_p \in \mathbb{R}^{(H_z W_z)\times H_x \times W_x}$ is the correlation result. Each similarity map of f_p represents the similarity between the corresponding pixel in the template feature maps and all pixels in the RoI feature maps. The most similar regions will be highlighted in each map, and the entire set of similarity maps provides prior information on the outline of the target [41]. Unlike traditional cross-correlation [1], pixel-wise correlation separately highlights different parts of the target in different similarity maps. Thus, regions corresponding to the top-left and bottom-right corners can be highlighted in some of the similarity maps. Although the corners are sometimes outside the target, the prior information of the target outline can help guide the learning of the spatial attention maps, which highlight informative regions corresponding to the corners.

After pixel-wise correlation, we separately learn spatial attention maps for each corner via two hourglass-like structures [33]. (An hourglass structure first downsamples features to capture global information and increase the receptive field, then upsamples them to increase the resolution.) The network learns spatial attention maps from the pixel-wise similarity maps, which adaptively attend to the top-left and bottom-right corners. The spatial attention maps are obtained by applying a sigmoid activation function to normalize the network output:

$$A_s^i = \sigma(\mathcal{H}^i(f_p)) \tag{6}$$

where $i \in \{t, b\}$ is the index distinguishing the two corners, $\mathcal{H}$ denotes the hourglass network, and A_s is the spatial attention map.

After the channel-wise and spatial attentions are obtained, feature $\phi_g(x)$ of the RoI is enhanced by sequential multiplication with the channel-wise attention descriptor and the spatial attention maps. The enhanced features are separately given by

$$\phi_e^t(x) = \phi_g(x) \otimes A_c \otimes A_s^t, \tag{7}$$

$$\phi_e^b(x) = \phi_g(x) \otimes A_c \otimes A_s^b, \tag{8}$$

where $\otimes$ denotes broadcasting element-wise multiplication, and $\phi_e^t(x)$ and $\phi_e^b(x)$ denote the features for the detection of the top-left and bottom-right corners, respectively.

Our version of correlation-guided attention has two advantages. First, target-specific information is incorporated to enhance features, which improves discriminative power. Second, corner regions are highlighted, which helps to detect the corners accurately.

Corner detection: After feature enhancement, we detect the target corners by predicting a heatmap for each corner and then calculating the corner coordinates by a Soft-argmax function [29]. The corner detection network employs an upsample structure to learn heatmaps for the two corners. (As the network architecture is the same for the two corners, we will describe only one of them.) We use several convolutional networks and nearest interpolation layers to continuously upsample the features and decrease the channel number. The heatmap for the corner is obtained from the output of the final convolutional layer. Then a Soft-argmax function is applied, which first normalizes the heatmap through a Softmax function and then calculates the expected value. The normalized heatmap can be viewed as a map of the probability that the corner is at position (x, y). The expected value of the position of the corner is obtained by

$$\hat{\mathbf{p}} = (\sum_{m=1}^{W_h}\sum_{n=1}^{H_h} m h_{n,m}, \sum_{n=1}^{H_h}\sum_{m=1}^{W_h} n h_{n,m})^T, \tag{9}$$

where h is the normalized heatmap of size $W_h \times H_h$ and $\hat{\mathbf{p}} = (\hat{p}_x, \hat{p}_y)$ is the position of the corner. The Soft-argmax function enables efficient sub-pixel localization. During training, we adopt the commonly used elastic net loss function [29] L_{cdet}.

3.3. Implementation

Network structure: We use ResNet-50 [14] pre-trained on the ImageNet dataset as our backbone network and extract generic feature representations from the last layer of the *conv4* block. The backbone is the same as in [23]. Detailed information about the network can be found in the supplementary material.

Training: The network is end-to-end trained by optimizing the loss function:

$$L = \lambda_1 L_{sim} + \lambda_2 L_{reg} + \lambda_3 L_{cdet}, \tag{10}$$

where λ_1, λ_2 and λ_3 are respectively set to 1, 1 and 0.125 to balance the three losses. The shrunk factors α_p and α_n are empirically set to 0.3 and 0.4.

The SGD optimizer is used to train the network. There are a total of 20 epochs. We use a warmup strategy with the learning rate increasing from 0.0005 to 0.001 in the first 5 epochs. In order to initialize the Siamese tracking module, only parameters in this module are trained in the first epoch. For the last 15 epochs, the learning rate decays exponentially from 0.001 to 0.0001. A weight decay of 0.0005 and momentum of 0.9 are used. During the first 10 epochs, the parameters of the backbone are frozen; during the last 10 epochs, the *conv4* and *conv3* blocks are unfrozen and fine-tuned with a learning rate 10 times smaller than other parameters.

We sample image pairs from the Youtube-BB [34], VID [35] and GOT-10k [18] datasets to train our network. Some still images are sampled from the COCO [26] and DET [35] datasets, as in [46]. To take into account a variety of target changes, we add various data augmentations, including random color jittering, random grayscale conversion, random translation within 64 pixels, and random resizing in the range [0.82, 1.18].

RoI augmentation During training, we construct several RoIs based on the results of the Siamese tracking module, which predicts one bounding box in each positively defined cell. During the training procedure, the target is ordinarily located in the center of the RoI. The result of the corner detection module in this situation is reliable. However, in the tracking process, the target is not always in the center of the RoI, since the Siamese tracking module may be distracted. Thus, we propose an RoI augmentation strategy of randomly augmenting half of the RoIs in every batch. Random translation within 8 pixels and resizing in the range [0.9, 1.1] are applied. This strategy effectually improves corner detection accuracy (see section 4.1).

Tracking: During tracking, the template image is cropped from the first frame, and the features extracted from the template image are fixed during the whole tracking process. As in [24], a cosine window and a scale change penalty are employed to select the best proposal from the output of the Siamese tracking module. This proposal is used to construct one RoI for corner detection in each frame. The tracking result is obtained from the output of the correlation-guided attentional corner detection module.

4. Experiments

In this section, we perform a detailed analysis of the proposed tracker and compare our results with state-of-the-art trackers on OTB2015 [43], VOT2018 [21], UAV123 [30], LaSOT [10], and TrackingNet [31]. Our tracker is implemented in Python using PyTorch. On a PC with a 3.5 GHz CPU and Nvidia RTX 2080Ti GPU, the proposed tracker achieves an average tracking speed of 70 FPS. The code will be made publicly available.

Method	Integration method	PS (%)	AUC (%)
W/o template	-	81.8	62.0
ConcatInte	concatenation	83.1	62.5
SiamInte	cross-correlation	85.3	64.9
ConcatAtt	concatenation attention	84.0	63.7
SiamAtt	cross-correlation attention	85.2	65.1
ChannelAtt	our channel attention	83.8	63.5
SpatialAtt	our spatial attention	85.4	65.3
Ours	our full attention	**86.7**	**66.3**

Table 1. Comparison of different approaches to integrating template and test image on the combined OTB2015 and UAV123 datasets. Our proposed template-guided attention module achieves the best results.

4.1. Method analysis

Here, we describe extensive experiments to analyze the impact of different components in the proposed tracker. Experiments are conducted on the combined OTB2015 [43] and UAV123 [30] datasets. There are a total of 223 challenging videos in this combined dataset, which enables a thorough method analysis. We use the precision score (PS) and area-under-the-curve (AUC) score [43] as evaluation metrics to compare different configurations of the proposed tracker.

Exploitation of Relationship: Our goal is to exploit the relationship between the template and the RoI to improve detection accuracy. We achieve this by employing our correlation-guided attention module as the integration method. We compare the proposed attention module with other integration methods to show its effectiveness. The results of different integration methods are shown in table 1. Except for the integration method, other components are kept the same.

First, we consider the case where integration of the template is not performed at all; compared to our method, performance drops by 4.9% in terms of PS and 4.3% in terms of AUC. This indicates the importance of relationship exploitation. We also compare our method with concatenation and cross-correlation, the integration methods respectively used in [15] and [12]. In table 1, *ConcatInte* represents the case in which we concatenate features from the template and the constructed RoI in the channel dimension; the template is cropped to a size similar to that of the RoI, and the feature maps of both have the same resolution. *SiamInte* represents the case in which we use a channel-wise cross-correlation operation to integrate the template and the RoI. In both of these methods, the features lack sufficient discriminative power, and spatial information about corners is not explicitly explored; consequently, both show inferior performances compared to our method.

Structure of attention module: The proposed attention module involves both spatial and channel-wise attention, learned from the pixel-correlation and channel-wise correlation results, respectively. We now compare our method with different structures to show the effectiveness of our attention module. As concatenation and cross-correlation by themselves do not achieve high performance, we compare our method to two alternative methods that respectively use concatenation and cross-correlation results to guide spatial and channel attention learning, as in our own method. As shown in table 1, *ConcatAtt* improves the *ConcatInte* by 0.9% in PS and 1.2% in AUC. However, *ConcatAtt* is still inferior to our method by 2.7% and 2.6% in PS and AUC, respectively. Similarly, *SiamAtt* achieves a slight performance improvement over *SiamInte*, while being outperformed by our method: 1.5% lower in PS and 1.2% lower in AUC. These results validate our choice of structure for the correlation-guided attention module.

To analyze the combined contribution of our channel and spatial modules, we show the results of using only channel-wise attention (*ChannelAtt*) and only spatial attention (*SpatialAtt*). The results in table 1 show that *SpatialAtt* achieves better performance (85.4% and 65.3%) than *ChannelAtt* (83.8% and 63.5%), and that they both improve the performance of *W/o template*. These results indicate that both spatial and channel-wise attention play significant roles in our method, but that spatial attention is the more important of the two; this is reasonable since corner detection requires rich spatial information. The combination of channel-wise and spatial attention, however, leads to even better performance (86.7% and 66.3%).

Siamese tracking module: The Siamese tracking module can be solely trained without the corner detection module to achieve a performance of 82.7% in PS and 61.5% in AUC, which provides a good initial target state and achieves the goal of effectively distinguishing the target from background distractors. However, our corner detection module improves it by 4.0% in PS and 4.8% in AUC. The Siamese tracking module can be replaced by a more efficient model (SiamFC [1]). Without the regression branch, we use the target size in the previous frame to construct the RoI during tracking. The resulting tracker achieves 85.5% in PS and 64.3% in AUC, a performance superior to that achieved with the baseline SiamFC (80.9% and 55.9%), which shows the effectiveness of our corner detection module.

Feature from different layers: We compare the performance of trackers based on features from different layers of the backbone network. Results in table 2 show that using features from *conv4* block achieves the best performance.

Impact of RoI augmentation: We propose that RoI augmentation can further improve tracking accuracy. The results in table 2 show that our tracker using RoI augmentation achieves a gain of 0.5% in PS and 0.5% in AUC over the un-augmented case.

	Conv3	*Conv4*	*Conv5*	RoI Aug. (*Conv4*)
PS (%)	83.5	86.7	83.8	**87.2**
AUC (%)	63.8	66.3	62.9	**66.8**

Table 2. Comparison of features from different layers on the combined OTB2015 and UAV123 datasets. Using features from the *conv4* block, our method obtains the best result, which is further improved by our RoI augmentation strategy.

	SATIN [12]	UPDT [3]	DaSiamRPN [46]	MFT [21]	LADCF [45]	ATOM [6]	SiamRPN++ [23]	SiamMask [40]	DiMP-50 [2]	**CGACD**
EAO↑	0.282	0.378	0.383	0.385	0.389	0.401	0.414	0.423	0.440	0.449
Accuracy↑	0.490	0.536	0.586	0.505	0.503	0.590	0.600	0.615	0.597	0.615
Robustness↓	-	0.184	0.276	0.140	0.159	0.204	0.234	0.248	0.153	0.173

Table 3. Comparison with state-of-the-art trackers on VOT2018. Our approach achieves the best EAO and accuracy.

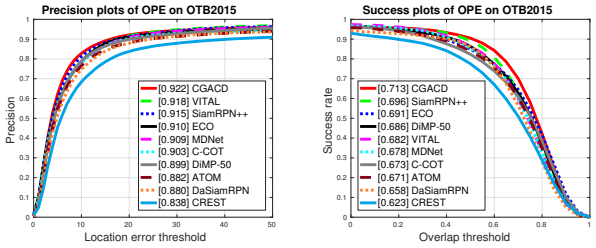

Figure 2. Comparison results of trackers on OTB2015.

In summary, the above experiments verify the effectiveness of our proposed architecture.

4.2. State-of-the-art comparison

We compare the proposed Correlation-Guided Attentional Corner Detection-based tracker (CGACD) to state-of-the-art trackers on five challenging tracking datasets.

VOT2018 dataset [21]: This dataset includes 60 videos for evaluating short-term object trackers in terms of accuracy (average overlap), robustness (average number of tracking failures), and expected average overlap (EAO), a combination of the other two metrics that is used to rank the trackers. We show the comparison results in table 3. Our CGACD achieves the best EAO (0.449) and accuracy (0.615). Compared with the best tracker in the VOT2018 challenge (LADCF [45]), CGACD achieves a performance gain of 6.0%. The superior accuracy of CGACD relative to ATOM [6] and SiamRPN++ [23] (which estimate bounding boxes but do not detect corners) shows the importance of corner detection. SATIN [12] does detect the corners of the target, but its performance is significantly lower than that of our tracker. Finally, although CGACD does not perform online updates, it can still achieve a robustness comparable to that of methods relying on online adaptation (MFT [21], LADCF [45], ATOM [6], and DiMP-50 [2]).

OTB2015 dataset [43]: This dataset consists of 100 videos; performance is evaluated using PS and AUC s-

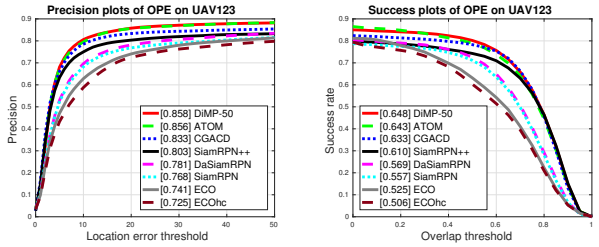

Figure 3. Comparison results of trackers on UAV123.

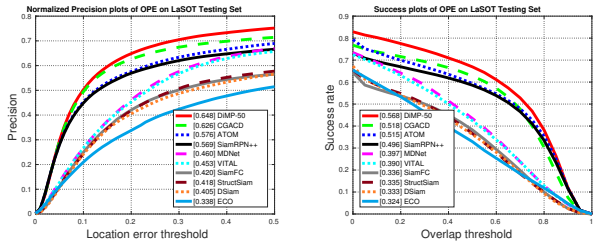

Figure 4. Comparison results of trackers on LaSOT.

cores. Figure 2 displays the comparison results. Our CGACD achieves the best performance in both metrics (92.2% and 71.3%). CGACD outperforms recent top-performer SiamRPN++ [23] by 0.7% in PS and 1.7% in AUC; ATOM [6] by 4.0% in PS and 4.2% in AUC, and DiMP-50 [2] by 2.3% in PS and 2.7% in AUC.

UAV123 dataset [30]: This dataset consists of 123 videos captured by unmanned aerial vehicles; the average sequence length is 915 frames. As in OTB2015, PS and AUC scores are employed to evaluate different trackers, but the videos in UAV123 are longer. Figure 3 shows the comparison results. Our CGACD achieves a PS of 83.3% and an AUC of 63.3%. Compared with the recent state-of-the-art tracker SiamRPN++ [23], we achieve a performance gain of 3.0% in PS and 2.3% in AUC. Although our tracker lacks online adaptation, it still achieves a performance comparable to that of ATOM [6] or DiMP-50 [2].

LaSOT dataset [10]: This is a recent large-scale and high-quality dataset. The average sequence length of a video is more than 2500 frames, which is very challenging for short-term trackers. In figure 4, we show normalized precision and success plots for 280 videos in the testing set. Our CGACD achieves 62.6% in normalized precision and 51.8% in AUC score. Our tracker outperforms the best tracker reported in the paper (MDNet [32]), by 16.6% and 12.1% in terms of normalized precision and AUC, respectively. Compared with SiamRPN++ [23], our tracker achieves a gain of 5.7% in normalized precision and 2.2% in AUC score. Although ATOM [6] employs online adaptation, it achieves a performance lower than that of our method by 5.0% in normalized precision and 0.3% in AUC. These results show that our method generalizes well to large-scale and challenging videos.

	ECO [5]	SiamFC [1]	GFS-DCF [44]	DaSiamRPN [46]	C-RPN [11]	SPM-Tracker [38]	ATOM [6]	SiamRPN++ [23]	DiMP-50 [2]	**CGACD**
PS (%)	49.2	53.3	56.6	59.1	61.9	66.1	64.8	69.4	68.7	69.3
PS_{norm} (%)	61.8	66.6	71.8	73.3	74.6	77.8	77.1	80.0	80.1	80.0
AUC (%)	55.4	57.1	60.9	63.8	66.9	71.2	70.3	73.3	74.0	71.1

Table 4. Comparison with state-of-the-art trackers on the TrackingNet test set.

TrackingNet dataset [31]: This large-scale dataset consists of real-world videos sampled from YouTube. We evaluate our tracker on the test set of 511 videos. Table 4 shows the results of our method compared with recent state-of-the-art trackers. Precision, normalized precision, and AUC scores are used to evaluate these trackers. Our CGACD obtains a precision of 69.3%, a normalized precision of 80.0, and an AUC score of 71.1%. The results achieved by our tracker are comparable with those achieved by other state-of-the-art trackers on all the metrics.

In summary, comparative testing on five datasets shows that our corner detection-based tracker achieves state-of-the-art performance.

5. Conclusion

We propose a novel corner detection-based tracking method that learns correlation-guided attention in a two-stage corner detection network. In the first stage, a light-weight Siamese network is exploited to distinguish the target from the background; in the second stage, it is used to construct an RoI for corner detection. The relationship between the template and the RoI is exploited to enhance features for corner detection by a correlation-guided attention module. The pixel-wise correlation result encoding the spatial information about the corners is exploited to guide the learning of spatial attention, while the channel-wise correlation result encoding the channel importance information is exploited to guide the learning of channel-wise attention. Target-specific information and spatial information about the corners are encoded in the enhanced features with the attention module, enabling accurate corner detection. The whole network is trained end-to-end on large-scale data sets, using an RoI augmentation strategy. Comprehensive experiments on five benchmark datasets show that the proposed tracker performs competitively with state-of-the-art trackers. In the future, we plan to explore efficient online adaptation method to improve the robustness of the proposed tracker.

Acknowledgement

This paper is supported in part by the National Natural Science Foundation of China (Grant No. 61671175), in part by the State Key Program of National Natural Science Foundation of China (Grant No. 51935005), and in part by the Sichuan Science and Technology Program (Grant No. 2019YFS0069).

References

[1] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshops*, 2016. 1, 2, 3, 4, 5, 7, 8

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *ICCV*, 2019. 7, 8

[3] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *ECCV*, 2018. 7

[4] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2

[5] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ECO: efficient convolution operators for tracking. In *CVPR*, 2017. 2, 8

[6] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. ATOM: accurate tracking by overlap maximization. In *CVPR*, 2019. 1, 2, 7, 8

[7] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 1, 2

[8] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 2

[9] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 2

[10] 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*, 2019. 2, 6, 8

[11] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019. 8

[12] Peng Gao, Yipeng Ma, Ruyue Yuan, Liyi Xiao, and Fei Wang. Siamese attentional keypoint network for high performance visual tracking. *arXiv preprint arXiv: 1904.10128*, 2019. 1, 2, 6, 7

[13] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 5

[15] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 FPS with deep regression networks. In *ECCV*, 2016. 1, 2, 6

[16] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015. 2

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 3, 4

[18] Lianghua Huang, Xin Zhao, and Kaiqi Huang. GOT-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv: 1810.11981*, 2018. 6

[19] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018. 4

[20] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, and Jianbo Shi. FoveaBox: Beyond anchor-based object detector. *arXiv preprint arXiv: 1904.03797*, 2019. 3, 4

[21] Matej Kristan et al. The sixth visual object tracking VOT2018 challenge results. In *ECCV Workshops*, 2018. 2, 6, 7

[22] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, pages 765–781, 2018. 1

[23] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. *arXiv preprint arXiv: 1812.11703*, 2018. 1, 2, 5, 7, 8

[24] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018. 1, 2, 4, 6

[25] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV*, 2014. 1, 2

[26] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 6

[27] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid R-CNN. In *CVPR*, 2019. 3, 4

[28] Alan Lukežič, Tomáš Vojíř, Luka Čehovin, Jiří Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017. 3

[29] Diogo C. Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. *Comput. Graph.*, 85:15–22, 2019. 3, 5

[30] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *ECCV*, 2016. 2, 6, 8

[31] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. TrackingNet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018. 2, 6, 8

[32] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 8

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 5

[34] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. YouTube-BoundingBoxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017. 6

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6

[36] Ming Tang, Bin Yu, Fan Zhang, and Jinqiao Wang. High-speed tracking with multi-kernel correlation filters. In *CVPR*, 2018. 2

[37] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 1, 2

[38] Guangting Wang, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. SPM-Tracker: series-parallel matching for real-time visual object tracking. In *CVPR*, 2019. 2, 8

[39] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen J. Maybank. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *CVPR*, 2018. 3

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 7

[41] Ziqin Wang, Jun Xu, Li Liu, Fan Zhu, and Ling Shao. Ranet: Ranking attention network for fast video object segmentation. In *ICCV*, 2019. 5

[42] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. In *ECCV Workshops*, 2018. 3, 4

[43] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *TPAMI*, 37(9):1834–1848, 2015. 2, 6, 7

[44] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Joint group feature selection and discriminative filter learning for robust visual object tracking. In *ICCV*, 2019. 8

[45] Tianyang Xu, Zhen-Hua Feng, Xiao-Jun Wu, and Josef Kittler. Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. *TIP*, 2019. 7

[46] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018. 1, 2, 6, 7, 8