High Performance Visual Tracking with Siamese Region Proposal Network

Bo Li[1,2], Junjie Yan[3],Wei Wu[1], Zheng Zhu[1,4,5], Xiaolin Hu[3]
[1] SenseTime Group Limited
[2] Beihang University
[3] Tsinghua University
[4] Institute of Automation, Chinese Academy of Sciences
[5] University of Chinese Academy of Sciences
{libo,wuwei}@sensetime.com yanjunjie@mail.tsinghua.edu.cn zhuzheng2014@ia.ac.cn

Abstract

Visual object tracking has been a fundamental topic in recent years and many deep learning based trackers have achieved state-of-the-art performance on multiple benchmarks. However, most of these trackers can hardly get top performance with real-time speed. In this paper, we propose the Siamese region proposal network (Siamese-RPN) which is end-to-end trained off-line with large-scale image pairs. Specifically, it consists of Siamese subnetwork for feature extraction and region proposal subnetwork including the classification branch and regression branch. In the inference phase, the proposed framework is formulated as a local one-shot detection task. We can pre-compute the template branch of the Siamese subnetwork and formulate the correlation layers as trivial convolution layers to perform online tracking. Benefit from the proposal refinement, traditional multi-scale test and online fine-tuning can be discarded. The Siamese-RPN runs at 160 FPS while achieving leading performance in VOT2015, VOT2016 and VOT2017 real-time challenges.

1. Introduction

Visual object tracking is a basic building block in various tasks of computer vision, such as automatic driving [19] and video surveillance [32]. It is challenging in large appearance variance caused by illumination, deformation, occlusion and motion [37, 39]. Besides, the speed is also important in practical applications [13, 4, 38].

Modern trackers can be roughly divided into two branches. The first branch is based on *correlation filter*, which trains a regressor by exploiting the properties of circular correlation and performing operations in the Fourier domain. It can do online tracking and update the weights of filters at the same time efficiently. The original version is conducted in Fourier domain and is then widely used in the tracking community [5, 14]. Recent correlation filter based methods use deep features to improve the accuracy, but it largely harms the speed during model update [10, 7]. Another branch of methods aims to use very strong deep features and do not update the model [13, 4, 35]. However, because the domain specific information is not used, performance of these methods is always not as good as correlation filter based methods.

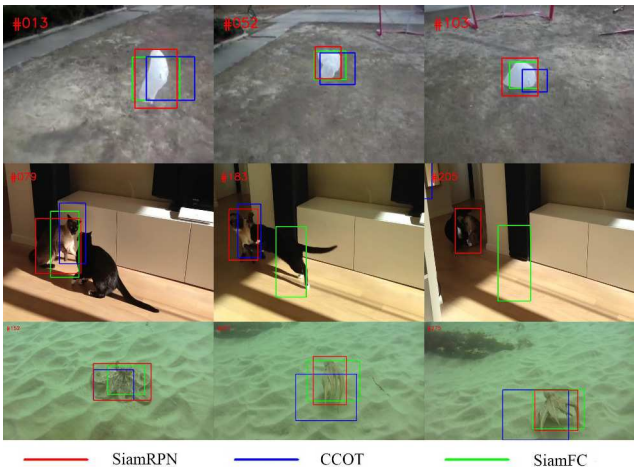

Figure 1: Comparisons of our approach with two state-of-the-art trackers. SiamRPN (short for Siamese-RPN) is able to predict the shape more precisely than SiamFC (short for Siamese-FC) [4], CCOT [10] when target's shape is severely changing.

In this paper, we show that the off-line trained deep learning based tracker can achieve competitive results compared to the state-of-the-art correlation filter based methods when properly designed. The key is the proposed Siamese region proposal network (Siamese-RPN). It consists of a template branch and a detection branch, which are trained off-line with large-scale image pairs in an end-to-end manner. Inspired by the state-of-the-art proposal extraction

method RPN [27], we perform proposal extraction on the correlation feature maps. Different from standard RPN, we use correlation feature map of the two branches for proposal extraction. In tracking task we don't have pre-defined categories, so we need the template branch to encode the target's appearance information into the RPN feature map to discriminate foreground from background.

For inference, we formulate it as a local one-shot detection framework, where the bounding box in the first frame is the only exemplar. We reinterpret the template branch as parameters to predict the detection kernels as a *meta-learner* like [2]. Both the meta-learner and the detection branch are trained end-to-end only using the RPN's supervision. The template branch is pruned to accelerate the speed after the initial frame during online tracking. To the best of our knowledge, this is the first work to formulate online tracking task as *one-shot detection*.

We evaluate the proposed method in VOT2015, VOT2016 and VOT2017 real-time challenges [17, 16, 15]. It can achieve leading performance in all of the three challenges. There are mainly two reasons why we can get state-of-the-art result without online fine-tuning. Firstly, our method can be trained off-line with image pairs, which can take advantage of the large-scale training data, such as Youtube-BB [25]. Ablation study shows that the more data can help to get even better performance. Secondly, We find that the the region proposal subnetwork usually predicts accurate scale and ratio of proposals to get compact bounding boxes as in Fig. 1.

The contributions can be summarized as three folds. 1). We propose the Siamese region proposal network (Siamese-RPN) which is end-to-end trained off-line with large-scale image pairs for the tracking task. 2). During online tracking, the proposed framework is formulated as a local one-shot detection task, which can refine the proposal to discard the expensive multi-scale test. 3). It achieves leading performance in VOT2015, VOT2016 and VOT2017 real-time challenges with the speed of 160 FPS, which proves its advantages in both accuracy and efficiency.

2. Related Works

Since the main contribution of this paper is the Siamese-RPN formulated as local one-shot detection task, we give a brief review on three aspects related to our work: trackers based on Siamese network structure, RPN in detection and one-shot learning.

2.1. Trackers based on Siamese network structure

A Siamese network consists of two branches which implicitly encodes the original patches to another space and then fuses them with a specific tensor to produce a single output. It's usually used for comparing two branches' features in the implicitly embedded space especially for contrastive tasks. Recently, Siamese networks have drawn great attention in visual tracking community because of their balanced accuracy and speed [13, 12, 4, 35, 36] . GOTURN [13] adopts the Siamese network as feature extractor and uses fully connected layers as the fusion tensor. It can be seen as a regression method by using predicted bounding box in the last frame as the only one proposal. Re3 [12] employs a recurrent network to get better feature produced by the template branch. Inspired by correlation based methods, Siamese-FC [4] first introduces the correlation layer as fusion tensor and highly improves the accuracy. The reason of its success is the densely supervised heatmap when comparing to GOTURN's one proposal regression, which enables Siamese-FC more robust to fast-moving objects. CFNet [35] adds a correlation filter to the template branch and makes the Siamese network shallower but more efficient. However, both Siamese-FC and CFNet are lack of bounding box regression and need to do multi-scale test which makes it less elegant. The main drawback of these real-time trackers is their unsatisfying accuracy and robustness compared to state-of-the-art correlation filter approaches.

2.2. RPN in detection

Region Proposal Network (RPN) is first proposed in Faster R-CNN [27]. Before RPN, traditional proposal extraction methods are time consuming. For example, Selective Search [34] needs 2 seconds to process one image. Besides, these proposals are not good enough for detection. The enumeration of multiple anchors [27] and sharing convolution features make the proposal extraction method time efficient while achieving high quality. RPN is capable of extracting more precise proposals due to the supervision of both foreground-background classification and bounding box regression. There are several variants of Faster R-CNN which employs RPN. R-FCN [6] takes component's position information into account and FPN [21] employs a feature pyramid network to improve the performance of tiny object detection. In contrast to two stage detectors, the improved versions of RPN, such as SSD [22] and YOLO9000 [26] are efficient detectors. RPN has many successful applications in detection because of its speed and great performance, however, it hasn't been fully exploited in tracking.

2.3. One-shot learning

In recent years, more and more attention has been paid to the one-shot learning topic in deep learning. Bayesian statistics based approaches and the meta-learning approaches are two major methods to solve the problem. In [20], object categories are represented by probabilistic models and Bayesian estimation is adopted in the inference phase. In another way, meta-learning approaches aim to get the ability of *learning to learn*, that is to say, being aware of and taking control of one's own learning. Concretely, [1] utilizes

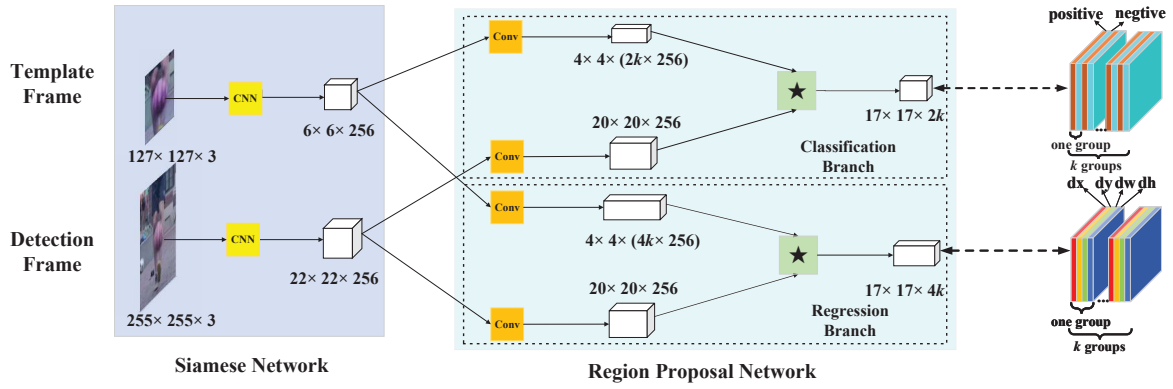

Figure 2: Main framework of Siamese-RPN: left side is Siamese subnetwork for feature extraction. Region proposal subnetwork lies in the middle, which has two branches, one for classification and the other for regression. Pair-wise correlation is adopted to obtain the output of two branches. Details of these two output feature maps are in the right side. In classification branch, the output feature map has $2k$ channels which corresponding to foreground and background of k anchors. In regression branch, the output feature map has $4k$ channels which corresponding to four coordinates used for proposal refinement of k anchors. In the figure, $\star$ denotes correlation operator.

a neural network to predict the gradient of the target network during the back-propagation. [30] learns a network that maps a small labelled support set and an unlabelled example to its label. Although these meta-learning based methods have got competitive results, these approaches are often evaluated on classification task and very few of them has extended to the tracking task. Learnet [2] is the first work to utilize the meta-learning method to solve the tracking task, which predicts the parameters of a pupil network from a single exemplar. However, the performance of Learnet is not competitive the modern DCF based methods, e.g. CCOT in multiple benchmarks.

3. Siamese-RPN framework

In this section, we describe the proposed Siamese-RPN framework in detail. As shown in Fig. 2, the proposed framework consists of a Siamese subnetwork for feature extraction and a region proposal subnetwork for proposal generation. Specifically, there are two branches in RPN subnetwork, one is in charge of the foreground-background classification, another is used for proposal refinement. Image patches including the target objects are fed into the proposed framework and the whole system is trained end-to-end.

3.1. Siamese feature extraction subnetwork

In Siamese network, we adopt a fully convolution network without padding. Let L_τ denote the translation operator $(L_\tau x)[u] = x[u - \tau]$, then all paddings are removed to satisfy the definition of fully convolution with stride k:

$$h(L_{k\tau}x) = L_\tau h(x) \tag{1}$$

Here we use the modified AlexNet [18], where the groups from conv2 and conv4 are removed [4].

The Siamese feature extraction subnetwork consists of two branches. One is called the *template branch* which receives target patch in the historical frame as input (denoted as z). The other is called the *detection branch* which receives target patch in the current frame as input (denoted as x). The two branches share parameters in CNN so that the two patches are implicitly encoded by the same transformation which is suitable for the subsequent tasks. For convenience, we denote $\varphi(z)$ and $\varphi(x)$ as the output feature maps of Siamese subnetwork.

3.2. Region proposal subnetwork

The region proposal subnetwork consists of a pair-wise correlation section and a supervision section. The supervision section has two branches, one for foreground-background classification and the other for proposal regression. If there are k anchors, network needs to output $2k$ channels for classification and $4k$ channels for regression. So the pair-wise correlation section first increase the channels of $\varphi(z)$ to two branches $[\varphi(z)]_{cls}$ and $[\varphi(z)]_{reg}$ which have $2k$ and $4k$ times in channel respectively by two convolution layers. $\varphi(x)$ is also split into two branches $[\varphi(x)]_{cls}$ and $[\varphi(x)]_{reg}$ by two convolution layers but keeping the channels unchanged. $[\varphi(z)]$ is served as the correlation kernel of $[\varphi(x)]$ in a "group" manner, that is to say, the channel number in a group of $[\varphi(z)]$ is the same as the overall channel number of $[\varphi(x)]$. The correlation is computed on both the classification branch and the regression branch:

$$\begin{aligned} A^{cls}_{w\times h\times 2k} &= [\varphi(x)]_{cls} \star [\varphi(z)]_{cls} \\ A^{reg}_{w\times h\times 4k} &= [\varphi(x)]_{reg} \star [\varphi(z)]_{reg} \end{aligned} \tag{2}$$

The template feature maps $[\varphi(z)]_{cls}$ and $[\varphi(z)]_{reg}$ are used as kernels and $\star$ denotes the convolution operation. As shown in Fig. 2, each point in $A^{cls}_{w\times h\times 2k}$ denoted as $(\tilde{w}, \tilde{h}, :)$ contains a $2k$ channel vector, which represents for negative and positive activation of each anchor at corresponding location on original map. Softmax loss is adopted to supervise the classification branch. Similarly, each point in $A^{reg}_{w\times h\times 4k}$ denoted as $(\hat{w}, \hat{h}, :)$ contains a $4k$ channel vector , which represents for dx, dy, dw, dh measuring the distance between anchor and corresponding groundtruth.

When training the network with several anchors, we employ the loss function that is used in Faster R-CNN [27]. Loss for classification is the cross-entropy loss and we adopt smooth L_1 loss with normalized coordinates for regression. Let A_x, A_y, A_w, A_h denote center point and shape of the anchor boxes and let T_x, T_y, T_w, T_h denote those of the ground truth boxes, the normalized distance is:

$$\begin{aligned} \delta[0] &= \frac{T_x - A_x}{A_w}, \quad \delta[1] = \frac{T_y - A_y}{A_h} \\ \delta[2] &= ln\frac{T_w}{A_w}, \quad \delta[3] = ln\frac{T_h}{A_h} \end{aligned} \tag{3}$$

Then they pass through smooth L_1 loss which can be written as below,

$$smooth_{L_1}(x,\sigma) = \begin{cases} 0.5\sigma^2 x^2, & |x| < \frac{1}{\sigma^2} \\ |x| - \frac{1}{2\sigma^2}, & |x| \geq \frac{1}{\sigma^2} \end{cases} \tag{4}$$

Finally we optimize the loss function

$$loss = L_{cls} + \lambda L_{reg} \tag{5}$$

where λ is hyper-parameter to balance the two parts. L_{cls} is the cross entropy loss and L_{reg} is:

$$L_{reg} = \sum_{i=0}^{3} smooth_{L1}(\delta[i], \sigma) \tag{6}$$

3.3. Training phase: End-to-end train Siamese-RPN

During the training phase, sample pairs are picked from ILSVRC [29] with a random interval and from Youtube-BB [25] continuously. The template and the detection patches are extracted from two frames of the same video. We train Siamese-RPN end-to-end using Stochastic Gradient Descent (SGD) after the Siamese subnetwork being pretrained using Imagenet. Because of the need of training regression branch, some data augmentations are adopted including affine transformation.

We choose less anchors in tracking task than detection task by noticing that the same object in two adjacent frames won't change much. So only one scale with different ratios of anchor is adopted and the anchor ratios we adopted are $[0.33, 0.5, 1, 2, 3]$.

The strategy to pick positive and negative training samples is also important in our proposed framework. The criterion used in object detection task is adopted here that we use IoU together with two thresholds th_{hi} and th_{lo} as the measurement. Positive samples are defined as the anchors which have $IoU > th_{hi}$ with their corresponding ground truth. Negative ones are defined as the anchors which satisfy $IoU < th_{lo}$. We set th_{lo} to 0.3 and th_{hi} to 0.6. We also limit at most 16 positive samples and totally 64 samples from one training pair.

4. Tracking as one-shot detection

In this subsection, we firstly formulate the tracking task as a local one-shot detection task. Afterwards, the inference phase under this interpretation is analyzed in detail and simplified to get a speed up. At last, some specific strategies are introduced to make the framework suitable for the tracking task.

4.1. Formulation

We consider one-shot detection as a discriminative task as in [2]. Its objective is to find the parameters W that minimize the average loss $\mathcal{L}$ of a predictor function $\psi(x; W)$. It is computed over a dataset of n samples x_i and corresponding labels ℓ_i:

$$\min_W \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\psi(x_i; W), \ell_i) \tag{7}$$

One-shot learning is aiming to learn W from a single template z of the class of interest. The challenge in discriminative one-shot learning is to find a mechanism to incorporate category information in the learner, i.e. *learning to learn*. To address the challenge, we propose a method to learn the parameters W of the predictor from a single template z using a meta-learning process, i.e. a feed-forward function ω that maps $(z; W^{'})$ to W. Let z_i be template samples in one batch then the problem can be formulated as:

$$\min_{W'} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\psi(x_i; \omega(z_i; W^{'})), \ell_i) \tag{8}$$

As the same above, let z denote for the template patch, x for the detection patch, function φ for the Siamese feature extraction subnetwork and function ζ for the region proposal subnetwork then the one-shot detection task can be formulated as:

$$\min_{W} \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\zeta(\varphi(x_i; W); \varphi(z_i; W)), \ell_i) \tag{9}$$

We can now reinterpret the template branch in Siamese subnetwork as training parameters to predict the kernel of the local detection task, which is typically the *learning to learn* process. In this interpretation, the template branch is used to embed the category information into the kernel and the detection branch performs detection using the embedded information. During the training phase, the meta-

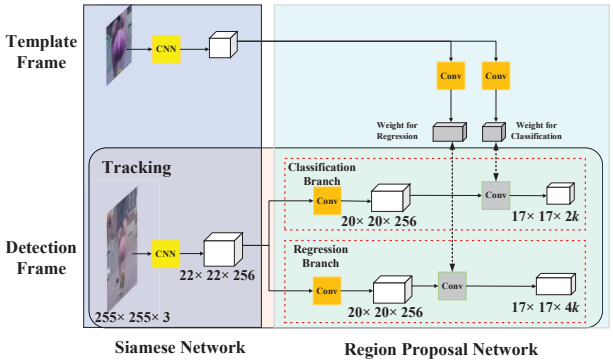

Figure 3: Tracking as one-shot detection: the template branch predicts the weights(in gray) for kernels of region proposal subnetwork on detection branch using the first frame. Then the template branch is pruned and only the detection branch is retained. So the framework is modified to a local detection network.

learner doesn't need any other supervision except the pairwise bounding box supervision. In the inference phase, Siamese framework is pruned only leaving the detection branch except the initial frame thus leading to high speed. The target patch from the first frame is sent into the template branch and the detection kernel is pre-computed so that we can perform one-shot detection in other frames. Because the local detection task is based on the category information only given by the template on initial frame, it can be viewed as one-shot detection.

4.2. Inference phase: Perform one-shot detection

As the formulation in Sec. 4.1, we regard the template branches' outputs as the kernels for local detection. Both the kernels are pre-computed on the initial frame and fixed during the whole tracking period. With the current feature map convolved by the pre-computed kernels, the detection branch performs online inference as one-shot detection as shown in Fig. 3. The forward pass on the detection branch is performed to obtain the classification and regression output, thus getting the top M proposals. Specifically, after the notation we defined in Eq. 2, we denote the classification and regression feature map as the point sets:

$$A^{cls}_{w\times h\times 2k} = \{(x^{cls}_i, y^{cls}_j, c^{cls}_l)\} \tag{10}$$

where $i \in [0, w)$, $j \in [0, h)$, $l \in [0, 2k)$.

$$A^{reg}_{w\times h\times 4k} = \{(x^{reg}_i, y^{reg}_j, dx^{reg}_p, dy^{reg}_p, dw^{reg}_p, dh^{reg}_p)\} \tag{11}$$

where $i \in [0, w)$, $j \in [0, h)$, $p \in [0, k)$.

Since the odd channels on the classification feature maps represent the positive activation, we collect the top K points in all $A^{cls}_{w\times h\times 2k}$ where l is odd number and denote the point set as $CLS^* = \{(x^{cls}_i, y^{cls}_j, c^{cls}_l)_{i\in I, j\in J, l\in L}\}$ where I, J, L are some index set. Variables i and j encode the location of corresponding anchor respectively, and l encode the ratio of corresponding anchor, so we can derive the corresponding anchor set as $ANC^* = \{(x^{an}_i, y^{an}_j, w^{an}_l, h^{an}_l)_{i\in I, j\in J, l\in L}\}$. Moreover, we find the activation of ANC^* on $A^{cls}_{w\times h\times 4k}$ to get the corresponding refinement coordinates as $REG^* = \{(x^{reg}_i, y^{reg}_j, dx^{reg}_l, dy^{reg}_l, dw^{reg}_l, dh^{reg}_l)_{i\in I, j\in J, l\in L}\}$. Afterwards, the refined top K proposals set $PRO^* = \{(x^{pro}_i, y^{pro}_j, w^{pro}_l, h^{pro}_l)\}$ can be obtained by following equations Eq. 12 :

$$\begin{aligned} x^{pro}_i &= x^{an}_i + dx^{reg}_l * w^{an}_l \\ y^{pro}_j &= y^{an}_j + dy^{reg}_l * h^{an}_l \\ w^{pro}_l &= w^{an}_l * e^{dw_l} \\ h^{pro}_l &= h^{an}_l * e^{dh_l} \end{aligned} \tag{12}$$

After the top K proposals are generated, we use some proposal selection strategy to make them suitable for the tracking task and we will discuss it in the next section.

4.3. Proposal selection

To make the one-shot detection framework suitable for tracking task, we propose two strategies to select the proposals.

The first proposal selection strategy is discarding the bounding boxes generated by the anchors too far away from the center. For example, we only keep the center $g \times g$ subregion on the $A^{cls}_{w\times h\times 2k}$ classification feature map to get $g \times g \times k$ anchors instead of $m \times n \times k$ anchors. Because the nearby frames always don't have large motion, the discard strategy can efficiently remove the outliers. Fig. 4 is a illustration of choosing target anchors whose distances are no more than 7 from the center in the classification feature map.

The second proposal selection strategy is that we use cosine window and scale change penalty to re-rank the proposals' score to get the best one. After the outliers are discarded, a cosine window is added to suppress the large displacement and then a penalty is added to suppress large change in size and ratio:

$$penalty = e^{k*max(\frac{r}{r'}, \frac{r'}{r})*max(\frac{s}{s'}, \frac{s'}{s})} \tag{13}$$

Here k is a hyper-parameter. r represents the proposal's ratio of height and width and r' represents that of last frame. s and s' represent the overall scale of the proposal and last frame, which is computed as below:

$$(w + p) \times (h + p) = s^2 \tag{14}$$

where w and h represent the width and height of the target, and p represents the padding which is equal to $\frac{w+h}{2}$. After these operations, the top K proposals are re-ranked after multiply the classification score by the temporal penalty. Non-maximum-suppression (NMS) is performed afterwards to get the final tracking bounding box. After the final bounding box is selected, target size is updated by linear interpolation to keep the shape changing smoothly.

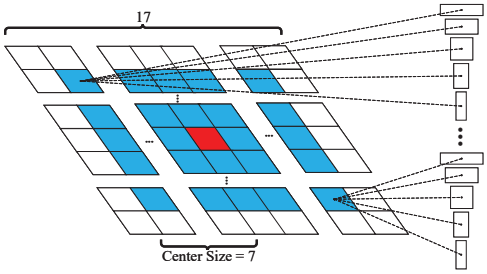

Figure 4: Illustration of center size 7 in RPN feature map, each grid represents encoded feature of k anchors at corresponding position. For example, there are $2k$ channels representing foreground and background activations in classification feature map. The center size of anchors indicate the search region of the model.

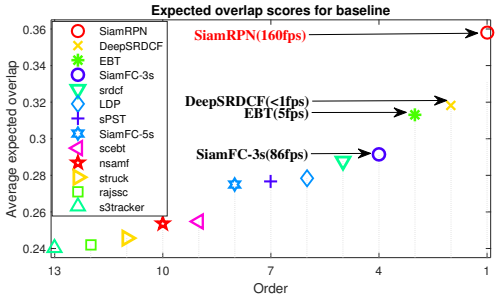

Figure 5: Expected overlap of our tracker, Siamese-FC and top 10 trackers in VOT2015 challenge.

5. Experiments

Experiments are performed on four challenging tracking datasets: VOT2015, VOT2016, VOT2017 real-time, each with 60 videos and OTB2015 with 100 videos. All the tracking results use the reported results to ensure a fair comparison.

5.1. Implementation details

We use a modified AlexNet pretrained from ImageNet [28] with the parameters of the first three convolution layers fixed and only fine-tune the last two convolution layers in Siamese-RPN. These parameters are obtained by optimizing the loss function in Eq. 5 with SGD. There are totally 50 epoches performed and the learning rate is decreased in log space from 10^{-2} to 10^{-6}. We extract image pairs from VID and Youtube-BB by choosing frames with interval less than 100 and performing further crop procedure. If the size of target's bounding box is denoted as (w, h), we crop the template patch centering on the historical frame with size $A \times A$ which is defined as follows.

$$(w + p) \times (h + p) = A^2 \tag{15}$$

where $p = \frac{w+h}{2}$. It is resized to 127×127 afterwards. In the same way the detection patch is cropped on the current frame with double the size of the template patch, and then resized in 255×255.

During inference phase, there is no online adaptation since we formulate online tracking as one-shot detection task. Our experiments are implemented using PyTorch on a PC with an Intel i7, 12G RAM, Nvidia GTX 1060.

5.2. Result on VOT2015

The VOT2015 dataset consists of 60 sequences. The performance is evaluated in terms of accuracy (average overlap while tracking successfully) and robustness (failure times). The overall performance is evaluated using Expected Average Overlap (EAO) which takes account of both accuracy and robustness. Besides, the speed is evaluated with a normalized speed (EFO).

Tracker	EAO	Accuracy	Failure	EFO
DeepSRDCF	*0.3181*	*0.56*	*1.0*	0.38
EBT	*0.313*	0.45	*1.02*	1.76
SRDCF	0.2877	0.55	1.18	1.99
LDP	0.2785	0.49	1.3	4.36
sPST	0.2767	0.54	1.42	1.01
SC-EBT	0.2548	0.54	1.72	0.8
NSAMF	0.2536	0.53	1.29	5.47
Struck	0.2458	0.46	1.5	2.44
RAJSSC	0.242	*0.57*	1.75	2.12
S3Tracker	0.2403	0.52	1.67	*14.27*
SiamFC-3s	0.2915	0.54	1.42	*8.68*
SiamFC-5s	0.275	0.53	1.45	7.84
SiamRPN	***0.358***	***0.58***	***0.93***	***23.0***

Table 1: Details about the state-of-the-art trackers in VOT2015. ***Red***, *blue* and *green*, represent *1st*, *2nd* and *3rd* respectively.

We compared our tracker with top 10 trackers according to the latest VOT rules (remove MDNet [24] from the board since it's trained with data generated from OTB's sequences). Siamese-FC is added into comparison as our baseline. Fig. 5 shows Siamese-RPN is able to outperform the trackers in VOT2015 and Tab. 1 lists the details about trackers. As shown in Tab. 1, Siamese-RPN is able to rank *1st* in EAO, accuracy, failure and EFO. Among all the trackers in VOT2015's report, only few trackers can track with real-time speed, but their expected overlap is relatively low. Siamese-FC is one of the top trackers on VOT2015 which can run at frame-rates beyond real-time and achieves state-of-the-art performance. Siamese-RPN is able to conduct at 160 FPS which is nearly two times of Siamese-FC (86 FPS), while gains 23% relative increase in EAO.

5.3. Result on VOT2016

In VOT2016 challenge, the sequences are the same as VOT2015, while the bounding boxes are re-annotated. The performance evaluations are the same as VOT2015.

We compare our tracker with top 25 trackers in VOT2016. Siamese-RPN can outperform all entries in challenge. Fig. 6 illustrates the EAO ranking and Tab. 2 shows

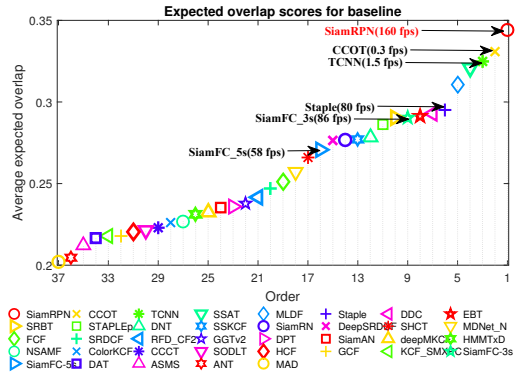

Figure 6: Expected overlap scores in the VOT2016 challenge, larger is better.

Tracker	EAO	Accuracy	Failure	EFO
C-COT	*0.331*	0.53	***0.85***	0.507
ECO-HC	*0.322*	0.53	*1.08*	*15.13*
Staple	0.2952	*0.54*	1.35	*11.14*
EBT	0.2913	0.47	*0.9*	3.011
MDNet_N	0.257	*0.54*	1.2	0.534
SiamRN	0.2766	*0.55*	1.37	5.44
SiamAN	0.2352	0.53	1.65	9.21
SiamRPN	***0.3441***	***0.56***	*1.08*	***23.3***

Table 2: Detail information about several published state-of-the-art trackers' performances in VOT2016.

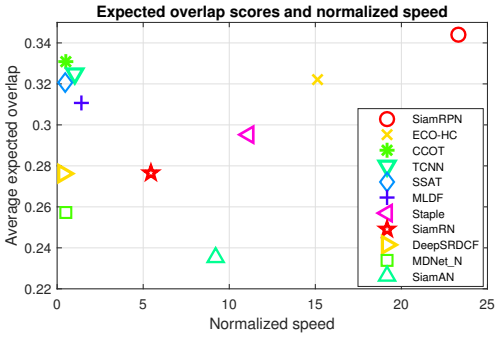

Figure 7: Performance and speed of our tracker and some state-of-the-art trackers in VOT2016. More closed to top means higher precision, and more closed to right means faster. Siamese-RPN is able to rank *1st* in EAO while operating at 160 FPS.

detail information about several state-of-the-art trackers. As shown in Fig. 6, our tracker can rank *1st* according to EAO while operating at 160 FPS, which is 500 times faster than CCOT. As shown in Tab. 2, Siamese-RPN ranks *1st* in EAO, accuracy and EFO, and *3rd* in failure. Fig. 7 shows the performance and speed of the state-of-the-art trackers. It shows our tracker can achieve a superior performance while operating at high speed.

5.4. Result on VOT2017 real-time experiment

In VOT2017 [15], the least 10 challenging sequences are replaced with 10 difficult sequences. Besides, a new real-time experiment is conducted, where trackers need to deal

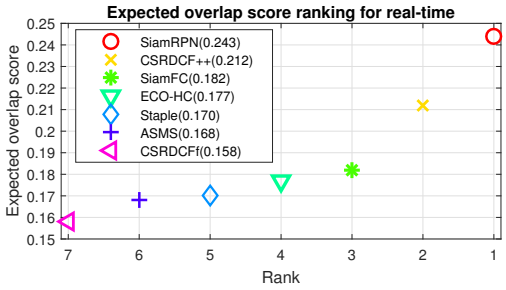

Figure 8: Expected overlap score ranking for the real-time experiment in the VOT2017 challenge.

with the real-time video stream at least 25 FPS. It means that if the tracker fails to process the result in 40 ms, the evaluator will use the bounding box of the last frame as the result of current frame. It is challenging for almost all of the state-of-the-art trackers. The top 10 trackers ranked by raw EAO under no speed limit criterion get lower EAO in real-time experiment.

Fig. 8 shows Siamese-RPN along with several real-time trackers listed in the report of the VOT2017. In comparison, Siamese-RPN can rank *1st* according to EAO. Specifically, it can surpass CSRDCF++ in the 2nd place by 14% and surpass Siamese-FC in the 3nd place by 33%.

5.5. Result on OTB2015

OTB2015 [37] contains 100 sequences that are collected from commonly used tracking sequences. The evaluation is based on two metrics: precision and success plot. The precision plot shows the percentage of frames that the tracking results are within 20 pixels from the target. The success plot shows the ratios of successful frames when the threshold varies from 0 to 1, where a successful frame means its overlap is larger than given threshold. The area under curve (AUC) of success plot is used to rank tracking algorithm.

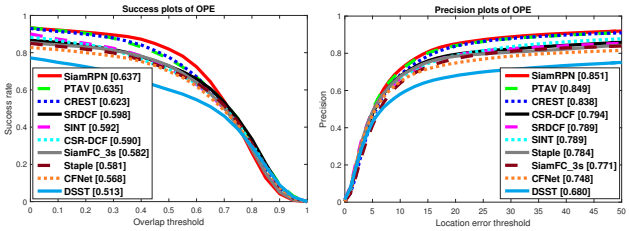

Figure 9: Success plot and precision plot of OTB2015

In this experiment, we compare our method with several representive trackers, including PTAV [11], CREST[31], SRDCF [8], SINT [33], CSR-DCF [23], Siamese-FC [4], Staple [3], CFNet [35] and DSST [9]. As shown in Fig. 9, the proposed Siamese-RPN is able to rank *1st* both in suc-

cess plot and precision plot.

5.6. Discussion

In this subsection, we discuss several factors that are essential to our performance, including data size, anchor ratios and positions.

5.6.1 Data size

Since our tracking framework only needs image pairs instead of continuous video streams, we are able to benefit from large-scale sparsely labelled videos. Compared to ILSVRC [29] which consists of about 4,000 videos annotated frame-by-frame, Youtube-BB [25] consists of more than 100,000 videos annotated once in every 30 frames. We train Siamese-RPN with different data set size by gradually adding more data from Youtube-BB.

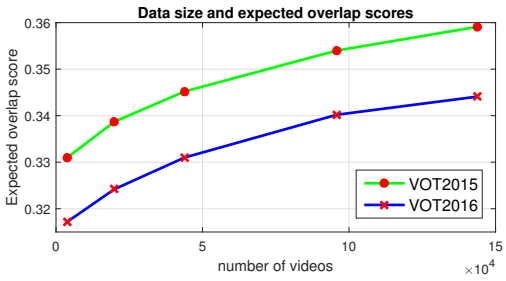

Figure 10: Effects of using increasing number of videos from ILSVRC [29] and Youtube-BB [25] on performance of tracker. Adding data from Youtube-BB can boost the performance gradually. Performance is not saturated, which means tracker performance may become better with more training data.

Fig. 10 illustrates tracking results of Siamese-RPN when the training data size varies. Both the EAO of VOT2015 and VOT2016 keeps increasing when there are more training videos. Specifically, the introducing of Youtube-BB boosts VOT2016's EAO from 0.317 to 0.344. It is noting that the performance is not saturated, which means tracker performance may become better with more training data.

5.6.2 Anchor selection

Here we will discuss two factors about anchors: anchor ratio selection during training and position selection during inference.

anchor ratios As discussed in Sec. 3.3, we only consider different ratios of anchors while fix scale of anchors since the target's scale won't change much in two adjacent frames. Three ratios are tried, $[0.5, 1, 2]$, $[0.33, 0.5, 1, 2, 3]$, $[0.25, 0.33, 0.5, 1, 2, 3, 4]$ (denoted as A_3, A_5, A_7, respectively).

As shown in Tab. 3, tracker with A_5 performs better than that with A_3, because it's easier to predict the shape of target with large ratio of height and width through more anchors. However, tracker with A_7 fails to keep improving performance, which we think may be caused by overfitting. When adding more training data from Youtube-BB, the EAO gap between A_7 and A_5 decreases from 0.013 to 0.007.

ratios	EAO(without Yuotube)	EAO(with Youtube)
A_3	0.279	0.311
A_5	0.317	0.344
A_7	0.304	0.337

Table 3: Anchor ratios and EAO on VOT2016. With/without Youtube means the model is trained with or without Youtube-BB, respectively.

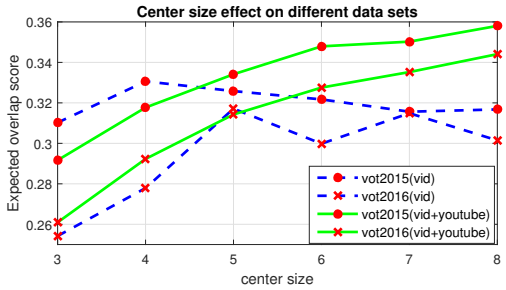

Figure 11: Center size effect on different data sets. Dashed lines and solid lines show the changes of model trained without and with Youtube-BB, respectively. Circle point and cross show the changes on VOT2015 and VOT2016, respectively. When adding Youtube-BB data set, the bigger center size we set, the better EAO we got. When using only VID data set, the best center size of anchor is 4, indicating that the discriminative ability of region proposal subnetwork is not good enough to use large search region.

anchor position In our experiment, center size (as defined in 4.3) is related to the size of search region. We can see that in Fig. 4, larger center size means tracker can pick anchors with a larger distance from the center to enlarge the search region. As shown in Fig. 11, when the network is trained with Youtube-BB, the performance becomes higher when the center size increases. However, if only trained with ILSVRC, the performance doesn't increase as expected, which means the discriminative ability of RPN is not good enough to use large search region.

6. Conclusion

In this work, we propose the Siamese region proposal network(Siamese-RPN) which is end-to-end offline trained with large-scale image pairs from ILSVRC and Youtube-BB. Siamese-RPN can get more accurate bounding boxes by applying box refinement procedure. During online tracking, the proposed framework is formulated as a local one-shot detection task. In experiments, our method can achieve leading performance in VOT2015, VOT2016 and VOT2017 real-time challenges while operating at 160 FPS.

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. *neural information processing systems*, pages 3981–3989, 2016.

[2] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, 2016.

[3] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865, 2016.

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition*, pages 2544–2550, 2010.

[6] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 379–387. Curran Associates, Inc., 2016.

[7] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[8] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[9] M. Danelljan, G. Hger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference*, pages 65.1–65.11, 2014.

[10] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488, 2016.

[11] H. Fan and H. Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[12] D. Gordon, A. Farhadi, and D. Fox. Re3 : Real-time recurrent regression networks for object tracking. *arXiv preprint arXiv:1705.06368*, 2017.

[13] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765, 2016.

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

[15] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, and G. Fernandez. The visual object tracking vot2017 challenge results. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[16] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. ehovin, T. Vojr, G. Hger, A. Lukei, and G. Fernndez. *The Visual Object Tracking VOT2016 Challenge Results*. Springer International Publishing, 2016.

[17] M. Kristan, J. Matas, A. Leonardis, and M. Felsberg. The visual object tracking vot2015 challenge results. In *IEEE International Conference on Computer Vision Workshop*, pages 564–586, 2016.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[19] K. H. Lee and J. N. Hwang. On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, 17(9):1429–1438, 2015.

[20] F. F. Li, R. Fergus, and P. Perona. *One-Shot Learning of Object Categories*. IEEE Computer Society, 2006.

[21] T. Y. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[23] A. Lukei, T. Voj, L. ehovin, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[25] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. *arXiv preprint arXiv:1702.00824*, 2017.

[26] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In *International Conference on Neural Information Processing Systems*, pages 91–99, 2015.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2014.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[30] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. Meta-learning with memory-augmented neu-

ral networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[31] Y. Song, C. Ma, L. Gong, J. Zhang, R. Lau, and M. H. Yang. Crest: Convolutional residual learning for visual tracking. *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[32] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person reidentification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548, 2017.

[33] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *Computer Vision and Pattern Recognition*, pages 1420–1429, 2016.

[34] J. R. R. Uijlings, K. E. A. V. D. Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[35] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. *The IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[36] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. Dcfnet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*, 2017.

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

[38] Z. Zhu, G. Huang, W. Zou, D. Du, and C. Huang. Uct: Learning unified convolutional networks for real-time visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Oct 2017.

[39] Z. Zhu, W. Wu, W. Zou, and J. Yan. End-to-end flow correlation tracking with spatial-temporal attention. *arXiv preprint arXiv:1711.01124*, 2017.