Siamese Cascaded Region Proposal Networks for Real-Time Visual Tracking

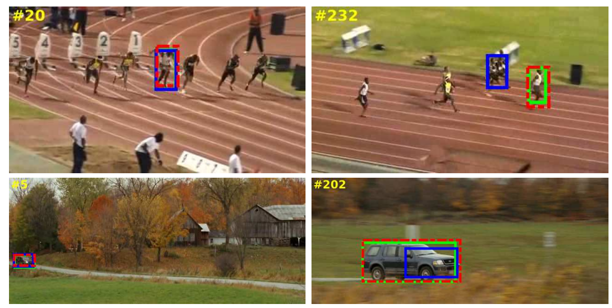
Heng Fan Haibin Ling*
Department of Computer and Information Sciences, Temple University, Philadelphia, PA USA
{hengfan,hbling}temple.edu

Abstract

Recently, the region proposal networks (RPN) have been combined with the Siamese network for tracking, and shown excellent accuracy with high efficiency. Nevertheless, previously proposed one-stage Siamese-RPN trackers degenerate in presence of similar distractors and large scale variation. Addressing these issues, we propose a multi-stage tracking framework, Siamese Cascaded RPN (C-RPN), which consists of a sequence of RPNs cascaded from deep high-level to shallow low-level layers in a Siamese network. Compared to previous solutions, C-RPN has several advantages: **(1)** *Each RPN is trained using the outputs of RPN in the previous stage. Such process stimulates hard negative sampling, resulting in more balanced training samples. Consequently, the RPNs are sequentially more discriminative in distinguishing difficult background (*i.e.*, similar distractors).* **(2)** *Multi-level features are fully leveraged through a novel feature transfer block (FTB) for each RPN, further improving the discriminability of C-RPN using both high-level semantic and low-level spatial information.* **(3)** *With multiple steps of regressions, C-RPN progressively refines the location and shape of the target in each RPN with adjusted anchor boxes in the previous stage, which makes localization more accurate. C-RPN is trained end-to-end with the multi-task loss function. In inference, C-RPN is deployed as it is, without any temporal adaption, for real-time tracking. In extensive experiments on OTB-2013, OTB-2015, VOT-2016, VOT-2017, LaSOT and TrackingNet, C-RPN consistently achieves state-of-the-art results and runs in real-time.*

1. Introduction

Visual tracking is one of the most fundamental problems in computer vision, and has a long list of applications such as robotics, human-machine interaction, intelligent vehicle, surveillance and so forth. Despite great advances in recent years, visual tracking remains challenging due to many factors including occlusion, scale variation, etc.



GT C-RPN Siamese-RPN

Figure 1. Comparisons between one-stage Siamese-RPN [23] and C-RPN on two challenging sequences: *Bolt2* (the top row) with similar distractors and *CarScale* (the bottom row) with large scale changes. We observe that C-RPN can distinguish the target from distractors, while Siamese-RPN drifts to the background in *Bolt2*. In addition, compared to using a single regressor in Siamese-RPN, multi-regression in C-RPN can better localize the target in presence of large scale changes in *CarScale*. Best viewed in color.

Recently, Siamese network has drawn great attention in the tracking community owing to its balanced accuracy and speed. By formulating object tracking as a matching problem, Siamese trackers [2, 17, 19, 23, 45, 46, 51, 59] aim to learn *offline* a generic similarity function from a large set of videos. Among these methods, the work of [23] proposes a one-stage Siamese-RPN for tracking by introducing the regional proposal networks (RPN), originally used for object detection [38], into Siamese network. With the proposal extraction by RPN, this approach simultaneously performs classification and localization from multiple scales, achieving excellent performance. Besides, the use of RPN avoids applying the time-consuming pyramid for target scale estimation [2], leading to a super real-time solution.

1.1. Problem and Motivation

Despite having achieved promising result, Siamese-RPN may drift to the background especially in presence of similar semantic distractors (see Fig. 1). We identify two reasons accounting for this.

First, the distribution of training samples is imbalanced:

*Corresponding author.

(1) positive samples are far less than negative samples, leading to ineffective training of the Siamese network; and (2) most negative samples are easy negatives (non-similar non-semantic background) that contribute *little* useful information in learning a discriminative classifier [29]. As a consequence, the classifier is dominated by the easily classified background samples, and degrades when encountering difficult similar semantic distractors.

Second, low-level spatial features are not fully explored. In Siamese-RPN (and other Siamese trackers), only features of the last layer, which contain more semantic information, are explored to distinguish target/background. In tracking, nevertheless, background distractors and the target may belong to the same category, and/or have similar semantic features [49]. In such case, the high-level semantic features are less discriminative in distinguishing target/background.

In addition to the issues above, the one-stage Siamese-RPN applies a single regressor for target localization using pre-defined anchor boxes. These anchors are expected to work well when having a high overlap with the target. However, for *model-free* visual tracking, no prior information regarding the target object is known, and it is hard to estimate how the scale of target changes. Using pre-defined coarse anchor boxes in a single step regression is insufficient for accurate localization [3, 14] (see again Fig. 1).

The class imbalance issue is addressed in two-stage object detectors (*e.g.*, Faster R-CNN [38]). The first proposal stage rapidly filters out most background samples, and then the second classification stage adopts sampling heuristics such as a fixed foreground-to-background ratio to maintain a manageable balance between foreground and background. In addition, two steps of regressions achieve accurate localization even for objects with extreme shapes.

Inspired by the two-stage detectors, we propose a multi-stage tracking framework by cascading a sequence of RPNs to solve the class imbalance problem, and meanwhile fully explore features across layers for robust visual tracking.

1.2. Contribution

As the **first contribution**, we introduce a novel multi-stage tracking framework, the Siamese Cascaded RPN (C-RPN), to solve the problem of class imbalance by performing hard negative sampling [40, 48]. C-RPN consists of a sequence of RPNs cascaded from the high-level to the low-level layers in the Siamese network. In each stage (level), an RPN performs classification and localization, and outputs the classification scores and the regression offsets for the anchor boxes in this stage. The easy negative anchors are then filtered out, and the rest, treated as hard examples, are used as training samples for the RPN of the next stage. Through such process, C-RPN performs stage by stage hard negative sampling. Consequently, the distributions of training samples are sequentially more balanced, and the classifiers of RPNs are sequentially more discriminative in distinguishing more difficult distractors (see Fig. 1).

Another benefit of C-RPN is more accurate target localization compared to the one-stage SiamRPN [23]. Instead of using the pre-defined coarse anchor boxes in a single regression step, C-RPN consists of multiple steps of regressions due to multiple RPNs. In each stage, the anchor boxes (including *locations* and *sizes*) are adjusted by the regressor, which provides better initialization for the regressor of next stage. As a result, C-RPN can progressively refine the target bounding box for better localization as shown in Fig. 1.

Leveraging features from different layers in the networks has been proven to be beneficial for improving model discriminability [27, 28, 30]. To fully explore both the high-level semantic and the low-level spatial features for visual tracking, we make the **second contribution** by designating a novel feature transfer block (FTB). Instead of separately using features from a single layer in one RPN, FTB enables us to fuse the high-level features into low-level RPN, which further improves its discriminative power to deal with complex background, resulting in better performance of C-RPN. Fig. 2 illustrates the framework of C-RPN.

In extensive experiments on six benchmarks, including OTB-2013 [52], OTB-2015 [53], VOT-2016 [20], VOT-2017 [21], LaSOT [10] and TrackingNet [34], our C-RPN achieves the state-of-the-art results and runs in real-time[1].

2. Related Work

Visual tracking has been extensively researched in recent decades. In the following we discuss the most related work, and refer readers to [25, 41, 54] for recent surveys.

Deep tracking. Inspired by the successes in image classification [18, 22], deep convolutional neural network (CNN) has been introduced into visual tracking and demonstrated excellent performances [7, 8, 12, 32, 35, 43, 49, 50]. Wang *et al.* [50] propose a stacked denoising autoencoder to learn generic feature representation for object appearance modeling in tracking. Wang *et al.* [49] present a fully convolutional network tracking approach by transferring the pre-trained deep features to improve tracking. Ma *et al.* [32] apply deep feature for correlation filter tracking, achieving remarkable gains. Nam and Han [35] propose a light architecture of CNN with online update to learn generic feature for tracking target. Fan and Ling [12] extend this approach by introducing a recurrent neural network (RNN) to capture object structure. Song *et al.* [43] apply adversary learning in CNN to learn richer representation for tracking. Danelljan *et al.* [8] propose continuous convolution filters for correlation filter tracking, and later optimize this method in [7].

Siamese tracking. Siamese network has attracted increas-

[1]The project is at http://www.dabi.temple.edu/~hbling/code/CRPN/crpn.htm

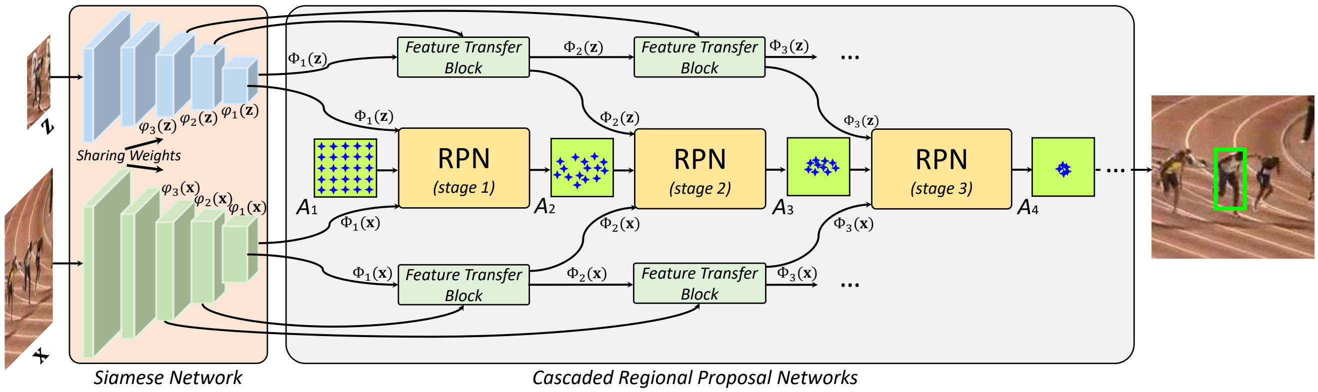


Figure 2. Illustration of the architecture of C-RPN, including a Siamese network for feature extraction and cascaded regional proposal networks for sequential classifications and regressions. The FTB transfers the high-level semantic features for the low-level RPN, and "A" represents the set of anchor boxes, which are gradually refined stage by stage. Best viewed in color.

ing interest for tracking because of its balanced accuracy and efficiency. Tao *et al.* [45] use Siamese network to learn a matching function from videos, then use the fixed matching function to search for the target. Bertinetto *et al.* [2] present a fully convolutional Siamese network (SiamFC) for tracking by measuring the region-wise feature similarity between the target and the candidate. Owing to its light structure and without model update, SiamFC runs efficiently at 80 fps. Held *et al.* [19] propose the GOTURN approach by learning a motion prediction model with the Siamese network. Valmadre *et al.* [46] use a Siamese network to learn the feature representation for correlation filter tracking. He *et al.* [17] introduce a two-fold Siamese network for tracking. Later in [16], they improve this two-fold Siamese tracking by incorporating angle estimation and spatial masking. Wang *et al.* [51] introduce an attention mechanism into Siamese network to learn a more discriminative metric for tracking. Notably, Li *et al.* [23] combine Siamese network with RPN and propose a one-stage Siamese-RPN tracker, achieving excellent performance. Zhu *et al.* [59] utilize more negative samples to improve the Siamese-RPN tracker. Despite improvement, this approach requires large extra training data from other domains.

Multi-level features. The features from different layers in the neural network contain different information. The high-level feature consists of more abstract semantic cues, while the low-level layers contains more detailed spatial information [30]. It has been proven that tracking can be benefited using multi-level features. In [32], Ma *et al.* separately use features in three different layers for three correlation models, and fuse their outputs for the final tracking result. Wang *et al.* [49] develop two regression models with features from two layers to distinguish similar semantic distractors.

Cascaded structure. Cascaded structures have been a popular strategy to improve performance. Viola *et al.* [48] propose a boosted cascade of simple feature for efficient object detection. Li *et al.* [24] present a cascaded structure built on CNN for face detection and achieve powerful discriminative capability with high efficiency. Cai *et al.* [3] propose a multi-stage object detection framework, cascade R-CNN, aiming at high quality detection by sequentially increasing IoU thresholds. Zhang *et al.* [55] utilize a cascade to refine detection results by adjusting anchors.

Our approach. In this paper, we focus on solving the problem of class imbalance to improve model discriminability. Our approach is related but different from the Siamese-RPN tracker [23], which applies one-stage RPN for classification and localization and skips the data imbalance problem. In contrast, our approach cascades a sequence of RPNs to address the data imbalance by performing hard negative sampling, and progressively refines anchor boxes for better target localization using multi-regression. Our method is also related to [32, 49] using multi-level features for tracking. However, unlike [32, 49] in which multi-level features are separately used for independent models (*i.e.*, *decision-level* fusion), we propose a feature transfer block to fuse the features across layers for each RPN (*i.e.*, *feature-level* fusion), improving its discriminative power in distinguishing the target object from complex background.

3. Siamese Cascaded RPN (C-RPN)

In this section, we detail the Siamese Cascaded RPN (referred to as C-RPN) as shown in Fig. 2.

C-RPN contains two subnetworks: the Siamese network and the cascaded RPN. The Siamese network is utilized to extract the features of the target template $\mathbf{x}$ and the search region $\mathbf{z}$. Afterwards, C-RPN receives the features of $\mathbf{x}$ and $\mathbf{z}$ for each RPN. Instead of only using the features from one layer, we apply feature transfer block (FTB) to fuse the features from high-level layers for RPN. An RPN simultaneously performs classification and localization on the feature maps of $\mathbf{z}$. Based on the classification scores and regression offsets, we filter out easy negative anchors (*e.g.*, an anchor

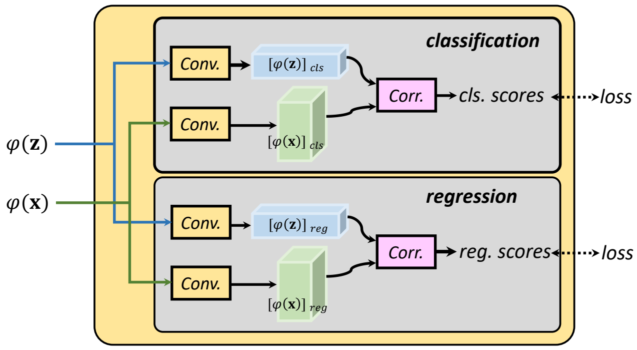


Figure 3. Architecture of RPN. Best viewed in color.

with negative confidence is larger than a preset threshold θ) and refine the rest for training RPN in the next stage.

3.1. Siamese Network

As in [2], we adopt the modified AlexNet [22] to develop our Siamese network. The Siamese network comprises two identical branches, the z-branch and the x-branch, which are employed to extract features from $\mathbf{z}$ and $\mathbf{x}$, respectively (see Fig. 2). The two branches are designed to share parameters to ensure the same transformation applied to both $\mathbf{z}$ and $\mathbf{x}$, which is crucial for the similarity metric learning. More details about the Siamese network can be referred to [2].

Different from [23] that only uses the features from the last layer of the Siamese network for tracking, we leverage the features from multiple levels to improve model robustness. For convenience in next, we denote $\varphi_i(\mathbf{z})$ and $\varphi_i(\mathbf{x})$ as the feature transformations of $\mathbf{z}$ and $\mathbf{x}$ from the conv-i layer in the Siamese network with N layers[2].

3.2. One-Stage RPN in Siamese Network

Before describing C-RPN, we first review the one-stage Siamese RPN tracker [23], which consists of two branches of classification and regression for anchors, as depicted in Fig. 3. It takes as inputs the feature transformations $\varphi_1(\mathbf{z})$ and $\varphi_1(\mathbf{x})$ of $\mathbf{z}$ and $\mathbf{x}$ and outputs classification scores and regression offsets for anchors. For simplicity, we remove the subscripts in feature transformations in next.

To ensure classification and regression for each anchor, two convolution layers are utilized to adjust the channels of $\varphi(\mathbf{z})$ into suitable forms, denoted as $[\varphi(\mathbf{z})]_{cls}$ and $[\varphi(\mathbf{z})]_{reg}$, for classification and regression, respectively. Likewise, we apply two convolution layers for $\varphi(\mathbf{x})$ but keep the channels unchanged, and obtain $[\varphi(\mathbf{x})]_{cls}$ and $[\varphi(\mathbf{x})]_{reg}$. Therefore, the classification scores $\{c_i\}$ and the regression offsets $\{r_i\}$

[2]For notation simplicity, we name each layer in the Siamese network in an **inverse** order, *i.e.*, conv-N, conv-$(N-1)$, $\cdots$, conv-2, conv-1 for the low-level to the high-level layers.

Figure 4. Localization using a single regressor and multiple regressors.The multiple regressors in C-RPN can better handle large scale changes for more accurate localization. Best viewed in color.

for each anchor can be computed as

$$\begin{aligned} \{c_i\} &= \mathrm{corr}([\varphi(\mathbf{z})]_{cls}, [\varphi(\mathbf{x})]_{cls}) \\ \{r_i\} &= \mathrm{corr}([\varphi(\mathbf{z})]_{reg}, [\varphi(\mathbf{x})]_{reg}) \end{aligned} \tag{1}$$

where i is the anchor index, and $\mathrm{corr}(\mathbf{a}, \mathbf{b})$ denotes correlation between $\mathbf{a}$ and $\mathbf{b}$ where $\mathbf{a}$ is served as the kernel. Each c_i is a 2d vector, representing for negative and positive confidences of the i^{th} anchor. Similarly, each r_i is a 4d vector which represents the offsets of center point location and size of the anchor to groundtruth. Siamese RPN is trained with a multi-task loss consisting of two parts, *i.e.*, the classification loss (*i.e.*, softmax loss) and the regression loss (*i.e.*, smooth L_1 loss). We refer readers to [23, 38] for further details.

3.3. Cascaded RPN

As mentioned earlier, previous Siamese trackers mostly ignore the problem of class imbalance, resulting in degenerated performance in presence of similar semantic distractors. Besides, they only use the high-level semantic features from the last layer, which does not fully explore multi-level features. To address these issues, we propose a multi-stage tracking framework by cascading a set of L $(L \leq N)$ RPNs.

For RPN$_l$ in the l^{th} $(1 < l \leq L)$ stage, it receives fused features $\Phi_l(\mathbf{z})$ and $\Phi_l(\mathbf{x})$ of conv-l layer and high-level layers from FTB. The $\Phi_l(\mathbf{z})$ and $\Phi_l(\mathbf{x})$ are obtained as follows,

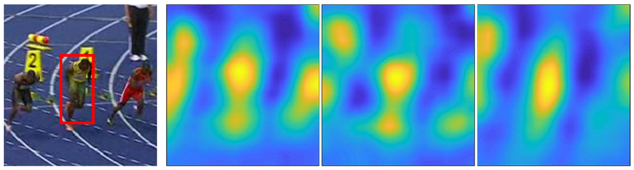
$$\begin{aligned} \Phi_l(\mathbf{z}) &= \mathrm{FTB}\big(\Phi_{l-1}(\mathbf{z}), \varphi_l(\mathbf{z})\big) \\ \Phi_l(\mathbf{x}) &= \mathrm{FTB}\big(\Phi_{l-1}(\mathbf{x}), \varphi_l(\mathbf{x})\big) \end{aligned} \tag{2}$$

where $\mathrm{FTB}(\cdot, \cdot)$ denotes FTB as described in Section 3.4. For RPN$_1$, $\Phi_1(\mathbf{z}) = \varphi_1(\mathbf{z})$ and $\Phi_1(\mathbf{x}) = \varphi_1(\mathbf{x})$. Therefore, the classification scores $\{c_i^l\}$ and the regression offsets $\{r_i^l\}$ for anchors in stage l are calculated as

$$\begin{aligned} \{c_i^l\} &= \mathrm{corr}([\Phi_l(\mathbf{z})]_{cls}, [\Phi_l(\mathbf{x})]_{cls}) \\ \{r_i^l\} &= \mathrm{corr}([\Phi_l(\mathbf{z})]_{reg}, [\Phi_l(\mathbf{x})]_{reg}) \end{aligned} \tag{3}$$

where $[\Phi_l(\mathbf{z})]_{cls}$, $[\Phi_l(\mathbf{x})]_{cls}$, $[\Phi_l(\mathbf{z})]_{reg}$ and $[\Phi_l(\mathbf{x})]_{reg}$ are derived by performing convolutions on $\Phi_l(\mathbf{z})$ and $\Phi_l(\mathbf{x})$.

Let A_l denote the anchor set in stage l. With classification scores $\{c_i^l\}$, we can filter out anchors in A_l whose negative confidences are larger than a preset threshold θ, and



(a) Region of interest (b) From left to right: response maps of stage 1, stage 2 and stage 3

Figure 5. Response maps in different stages. Image (a) is the region of interest, and (b) shows the response maps obtained by RPN in three stages. We can see that RPN is sequentially more discriminative in distinguishing distractors. Best viewed in color.

the rest are formed into a new set of anchor A_{l+1}, which is employed for training RPN_{l+1}. For RPN_1, A_1 is predefined. Besides, in order to provide a better initialization for regressor of RPN_{l+1}, we refine the anchors in A_{l+1} using regression results $\{r_i^l\}$ in RPN_l, thus generate more accurate localization compared to a single step regression in Siamese RPN [23], as illustrated in Fig. 4. Fig. 2 shows the cascade architecture of C-RPN.

The loss function ℓ_{RPN_l} for RPN_l is composed of classification loss function L_{cls} (softmax loss) and regression loss function L_{loc} (smooth L_1 loss) as follows,

$$\ell_{\text{RPN}_l}(\{c_i^l\}, \{r_i^l\}) = \sum_i L_{\text{cls}}(c_i^l, c_i^{l*}) + \lambda \sum_i c_i^{l*} L_{\text{loc}}(r_i^l, r_i^{l*}) \tag{4}$$

where i is the anchor index in A_l of stage l, λ a weight to balance losses, c_i^{l*} the label of anchor i, and r_i^{l*} the true distance between anchor i and groundtruth. Following [38], $r_i^{l*} = (r_{i(\text{x})}^{l*}, r_{i(\text{y})}^{l*}, r_{i(\text{w})}^{l*}, r_{i(\text{h})}^{l*})$ is a 4d vector, such that

$$\begin{aligned} r_{i(\text{x})}^{l*} &= (x^* - x_a^l)/w_a^l \quad & r_{i(\text{y})}^{l*} &= (y^* - y_a^l)/h_a^l \\ r_{i(\text{w})}^{l*} &= \log(w^*/w_a^l) \quad & r_{i(\text{h})}^{l*} &= \log(y^*/h_a^l) \end{aligned} \tag{5}$$

where x, y, w and h are center coordinates of a box and its width and height. Variables x^* and x_a^l are for groundtruth and anchor of stage l (likewise for y, w and h). It is worth noting that, different from [23] using fixed anchors, the anchors in C-RPN are progressively adjusted by the regressor in the previous stage, and computed as

$$\begin{aligned} x_a^l &= x_a^{l-1} + w_a^{l-1} r_{i(\text{x})}^{l-1} \quad & y_a^l &= y_a^{l-1} + h_a^{l-1} r_{i(\text{y})}^{l-1} \\ w_a^l &= w_a^{l-1} \exp(r_{i(\text{w})}^{l-1}) \quad & h_a^l &= h_a^{l-1} \exp(r_{i(\text{h})}^{l-1}) \end{aligned} \tag{6}$$

For the anchor in A_1, x_a^1, y_a^1, w_a^1 and h_a^1 are pre-defined.

The above procedure forms the proposed cascaded RPN. Due to the rejection of easy negative anchors, the distribution of training samples for each RPN is gradually more balanced. As a result, the classifier of each RPN is sequentially more discriminative in distinguishing difficult distractors. Besides, multi-level feature fusion further improves the discriminability in handing complex background. Fig. 5 shows the discriminative powers of different RPNs by demonstrating detection response map in each stage.

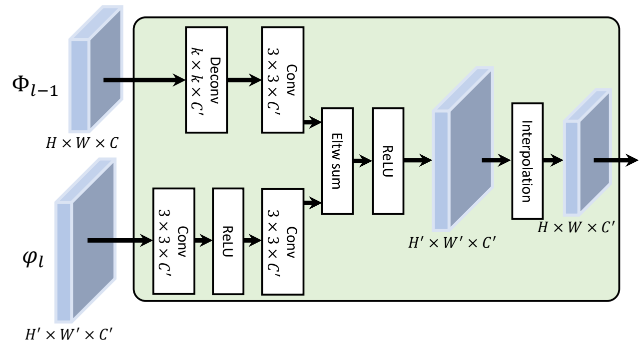


Figure 6. Overview of feature transfer block. Best viewed in color.

The loss function ℓ_{CRPN} of C-RPN consists of the loss functions of all RPN_l. For each RPN, loss function is computed using Eq. (4), and ℓ_{CRPN} is expressed as

$$\ell_{\text{CRPN}} = \sum_{l=1}^{L} \ell_{\text{RPN}_l} \tag{7}$$

3.4. Feature Transfer Block

To effectively leverage multi-level features, we introduce FTB to fuse features across layers so that each RPN is able to share high-level semantic feature to improve the discriminability. In detail, a deconvolution layer is used to match the feature dimensions of different sources. Then, different features are fused using element-wise summation, followed a ReLU layer. In order to ensure the same groundtruth for anchors in each RPN, we apply the interpolation to rescale the fused features such that the output classification and regression maps have the same resolution for all RPN. Fig. 6 shows the feature transferring for RPN_l ($l > 1$).

3.5. Training and Tracking

Training. The training of C-RPN is performed on the image pairs that are sampled from the same sequence as in [23]. The multi-task loss function in Eq. (7) enables us to train C-RPN in an end-to-end manner. Considering that the scale of target changes smoothly in two consecutive frames, we employ one scale with different ratios for each anchor. The ratios of anchors are set to $[0.33, 0.5, 1, 2, 3]$ as in [23].

For each RPN, we adopt the strategy as in object detection [38] to determine positive and negative training samples. We define the positive samples as anchors whose Intersection over union (IOU) with groundtruth is larger than a threshold τ_{pos}, and negative samples as anchors whose IoU with groundtruth bounding box is less than a threshold τ_{neg}. We generate at most 64 samples from one image pair.

Tracking. We formulate tracking as multi-stage detection. For each video, we pre-compute feature embeddings for the target template in the first frame. In a new frame, we extract a region of interest according to the result in last frame, and

Algorithm 1: Tracking with C-RPN

Input: frame sequences $\{\mathbf{X}_t\}_{t=1}^{T}$ and groundtruth bounding box $\mathbf{b}_1$ of $\mathbf{X}_1$, trained model C-RPN;
Output: Tracking results $\{\mathbf{b}_t\}_{t=2}^{T}$;
Extract target template $\mathbf{z}$ in $\mathbf{X}_1$ using $\mathbf{b}_1$;
Extract features $\{\varphi_l(\mathbf{z})\}_{l=1}^{L}$ for $\mathbf{z}$ from C-RPN;
for $t = 2$ to T **do**
  Extract the search region $\mathbf{x}$ in $\mathbf{X}_t$ using $\mathbf{b}_{t-1}$;
  Extract features $\{\varphi_l(\mathbf{x})\}_{l=1}^{L}$ for $\mathbf{x}$ from C-RPN;
  Initialize anchors A_1;
  for $l = 1$ to L **do**
    if l equals to 1 **then**
      $\Phi_l(\mathbf{z}) = \varphi_l(\mathbf{z})$, $\Phi_l(\mathbf{x}) = \varphi_l(\mathbf{x})$;
    else
      $\Phi_l(\mathbf{z})$, $\Phi_l(\mathbf{x}) \leftarrow$ Eq. (2) ;
    end
    $\{c_i^l\}$, $\{r_i^l\} \leftarrow$ Eq. (3);
    Remove any anchor i from A_l with negative confidence $c_{i(\text{neg})}^l > \theta$;
    $A_{l+1} \leftarrow$ Refine the rest anchors in A_l with $\{r_i^l\}$ using Eq. (6);
  end
  Target proposals $\leftarrow A_{L+1}$;
  Select the best proposal as tracking result $\mathbf{b}_k$ using strategies in [23];
end

then perform detection using C-RPN on this region. In each stage, an RPN outputs classification scores and regression offsets for anchors. The anchors with negative scores larger than θ are discarded, and the rest are refined and taken over by RPN in next stage. After the last stage L, the remained anchors are regarded as target proposals, from which we select the best one as the final tracking result using strategies in [23]. Alg. 1 summarizes the tracking process by C-RPN.

4. Experiments

Implementation detail. We implement C-RPN in Matlab using MatConvNet [47] on a single Nvidia GTX 1080 with 8GB memory. The backbone Siamese network adopts the modified AlexNet [22]. Instead of training from scratch, we borrow the parameters from the pretrained model on ImageNet [9]. During training, the parameters of first two layers are frozen. The number L of stages is 3. The thresholds θ, τ_{pos} and τ_{neg} are empirically set to 0.95, 0.6 and 0.3. C-RPN is trained end-to-end over 50 epochs using SGD, and the learning rate is annealed geometrically at each epoch from 10^{-2} to 10^{-6}. We train our C-RPN using the training data from [10] for experiments on LaSOT [10], and using VID [39] and YT-BB [37] for other experiments.

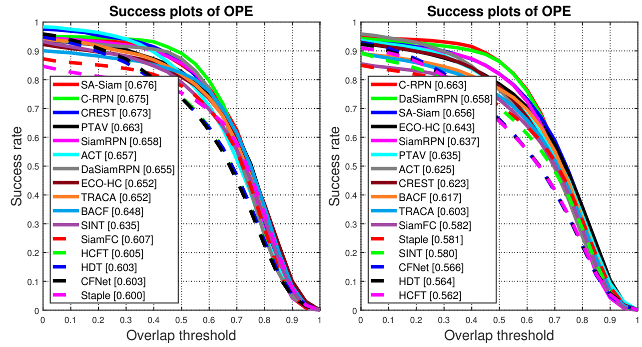


Figure 7. Comparisons with stage-of-the-art tracking approaches on OTB-2013 [52] and OTB-2015 [53]. C-RPN achieves the best results on both benchmarks. Best viewed in color.

4.1. Experiments on OTB-2013 and OTB-2015

We conduct experiments on the popular OTB-2013 [52] and OTB-2015 [53] which consist of 51 and 100 fully annotated videos, respectively. C-RPN runs at around 36 fps.

Following [52], we employ the *success* plot in *one-pass evaluation* (OPE) to assess different trackers. The comparison with 15 state-of-the-art trackers (SiamRPN [23], DaSiamRPN [59], TRACA [6], ACT [4], BACF [13], ECO-HC [7], CREST [42], SiamFC [2], Staple [1], PTAV [11], SINT [45], CFNet [46], SA-Siam [17], HDT [36] and HCFT [32]) is shown in Fig. 7. C-RPN achieves promising performance on both two benchmarks. In specific, we obtain the 0.675 and 0.663 precision scores on OTB-2013 and OTB-2015, respectively. In comparison with the baseline SiamRPN with 0.658 and 0.637 precision scores, we obtain improvements by 1.9% and 2.6%, showing the advantages of multi-stage RPNs in accurate localization. DaSiamRPN uses extra negative training data from other domains to improve the ability to handle similar distractors, and obtains 0.655 and 0.658 precision scores. Without using extra training data, C-RPN outperforms DaSiamRPN by 2.0% and 0.5%. More results and comparisons on OTB-2013 [52] and OTB-2015 [53] are shown in the supplementary material.

4.2. Experiments on VOT-2016 and VOT-2017

VOT-2016 [20] consists of 60 sequences, aiming at assessing the short-term performance of trackers. The overall performance of a tracking algorithm is evaluated using Expected Average Overlap (EAO) which takes both accuracy and robustness into account. The speed of a tracker is represented with a normalized speed (EFO).

We evaluate C-RPN on VOT-2016, and compare it with 11 trackers including the baseline SiamRPN [23] and other top ten approaches in VOT-2016. Fig. 8 shows the EAO of different trackers. C-RPN achieves the best results, significantly outperforming SiamRPN and other approaches. Tab. 1 lists the comparisons of different trackers on VOT-2016, and we can see that C-RPN outperforms other track-

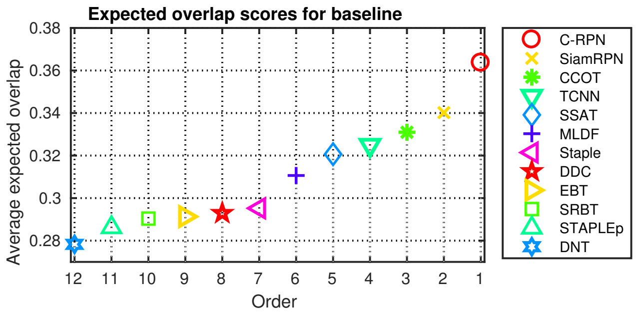


Figure 8. Comparisons on VOT-2016 [20]. Larger (right) value indicates better performance. Our C-RPN significantly outperforms the baseline and other approaches. Best viewed in color.

Table 1. Detailed comparisons on VOT-2016 [20]. The best two results are highlighted in **red** and **blue** fonts, respectively.

Tracker	EAO	Accuracy	Failure	EFO
C-RPN	**0.363**	**0.594**	0.95	9.3
SiamRPN [23]	**0.344**	0.560	1.12	**23.0**
C-COT [8]	0.331	0.539	**0.85**	0.5
TCNN [20]	0.325	0.554	0.96	1.1
SSAT [20]	0.321	**0.577**	1.04	0.5
MLDF [20]	0.311	0.490	**0.83**	1.2
Staple [1]	0.295	0.544	1.35	11.1
DDC [20]	0.293	0.541	1.23	0.2
EBT [58]	0.291	0.465	0.90	3.0
SRBT [20]	0.290	0.496	1.25	3.7
STAPLEp [20]	0.286	0.557	1.32	**44.8**
DNT [5]	0.278	0.515	1.18	1.1

ers in both accuracy and robustness, and runs efficiently.

VOT-2017 [21] contains 60 sequences, which are developed by replacing the 10 least challenging videos in VOT-2016 [20] with 10 difficult sequences. Different from VOT-2016 [20], VOT-2017 [21] introduces a new real-time experiment by taking into both tracking performance and efficiency. We compare C-RPN with SiamRPN [23] and other top ten approaches in VOT-2017 using the EAO of baseline and real-time experiments. As shown in Tab. 2, C-RPN achieves the EAO score of 0.289, which significantly outperforms SiamRPN [23] with 0.243 EAO score. Besides, compared to LSART [44] and CFWCR [21], C-RPN shows competitive performance. In real-time experiment, C-RPN obtains the best performance with EAO score of 0.273, outperforming all other trackers.

4.3. Experiment on LaSOT

LaSOT [10] is a recent large-scale dataset aiming at both training and evaluating trackers. There are 1,400 videos in LaSOT. We compare the proposed C-RPN to 35 approaches in LaSOT, including ECO [7], MDNet [35], SiamFC [2], VITAL [43], StructSiam [57], TRACA [6], BACF [13] and so on. We refer readers to [10] for more details about these trackers. In addition, we also compare C-RPN with the recent SiamRPN [23] tracker as it is an important baseline.

Following [10], we report the results of *success* (SUC) for different trackers as shown in Fig. 9. It shows that our C-RPN outperforms all other trackers under two protocols. We achieve SUC scores of 0.459 and 0.455 under protocol I and II, outperforming the second best tracker SiamRPN with SUC scores 0.44 and 0.433 by 1.9% and 2.2%, respectively. Compared to SiamFC with 0.358 and 0.336 SUC scores, C-RPN gains the improvements by 11.1% and 11.9%. C-RPN runs at around 23 fps on LaSOT. We refer readers to supplementary material for more details about results and comparisons on LaSOT.

Table 2. Comparisons on VOT-2017 [21]. The best two results are highlighted in **red** and **blue** fonts, respectively.

Tracker	Baseline EAO	Real-time EAO
C-RPN	0.289	**0.273**
SiamRPN [23]	0.243	**0.244**
LSART [44]	**0.323**	0.055
CFWCR [21]	**0.303**	0.062
CFCF [15]	0.286	0.059
ECO [7]	0.280	0.078
Gnet [21]	0.274	0.060
MCCT [21]	0.270	0.061
C-COT [8]	0.267	0.058
CSRDCF [31]	0.256	0.100
SiamDCF [21]	0.249	0.135
MCPF [56]	0.248	0.060

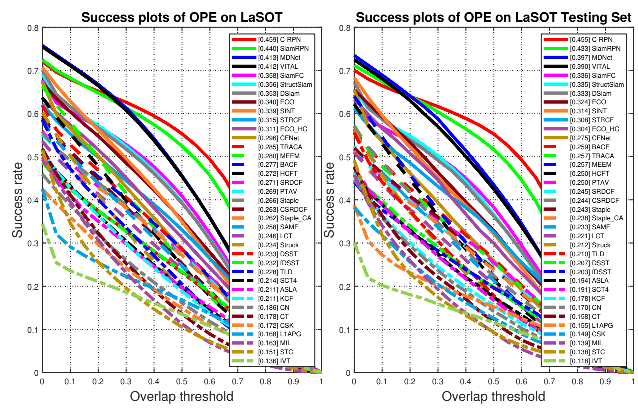


Figure 9. Comparisons with state-of-the-art tracking methods on LaSOT [10]. C-RPN outperforms existing approaches on success by large margins under all two protocols. Best viewed in color.

4.4. Experiment on TrackingNet

TrackingNet [34] is proposed to assess the performance of a tracker in the wild. We evaluate C-RPN on its testing set with 511 videos. Following [34], we utilize three metrics *precision* (PRE), *normalized precision* (NPRE) and *success* (SUC) for evaluation. Tab. 3 demonstrates the comparison

Table 3. Comparisons on TrackingNet [34] with the best two results highlighted in **red** and **blue** fonts, respectively.

	PRE	NPRE	SUC
C-RPN	**0.619**	**0.746**	**0.669**
MDNet [35]	**0.565**	**0.705**	**0.606**
CFNet [46]	0.533	0.654	0.578
SiamFC [2]	0.533	0.663	0.571
ECO [7]	0.492	0.618	0.554
CSRDCF [31]	0.48	0.622	0.534
SAMF [26]	0.477	0.598	0.504
ECO-HC [7]	0.476	0.608	0.541
Staple [1]	0.470	0.603	0.528
Staple_CA [33]	0.468	0.605	0.529
BACF [13]	0.461	0.580	0.523

Table 4. Effect on the number of stages in C-RPN.

# Stages	One stage	Two stages	Three stages
SUC on LaSOT	0.417	0.446	0.455
Speed on LaSOT	48 fps	37 fps	23 fps
EAO on VOT-2017	0.248	0.278	0.289

Table 5. Effect on negative anchor filtering (NAF) in C-RPN.

	C-RPN w/o NAF	C-RPN w/ NAF
SUC on LaSOT	0.439	0.455
EAO on VOT-2017	0.282	0.289

Table 6. Effect on feature transfer block in C-RPN. "S" and "M" indicate using single and multiple layers, respectively.

	C-RPN w/o FTB (S)	C-RPN w/o FTB (M)	C-RPN w/ FTB (M)
SUC on LaSOT	0.442	0.449	0.455
EAO on VOT-2017	0.278	0.282	0.289

results to trackers with top PRE scores[3], showing that C-RPN achieves the best results on all three metrics. In specific, C-RPN obtains the PRE score of 0.619, NPRE score of 0.746 and SUC score of 0.669, outperforming the second best tracker MDNet with PRE score of 0.565, NPRE score of 0.705 and SUC score of 0.606 by 5.4%, 4.1% and 6.3%, respectively. Besides, C-RPN runs efficiently at a speed of around 32 fps.

4.5. Ablation Experiment

To validate the impact of different components, we conduct ablation experiments on LaSOT (Protocol II) [10] and VOT-2017 [21].

Number of stages? As shown in Tab. 4, adding the second stage significantly improves one-stage baseline. The SUC on LaSOT is improved by 2.9% from 0.417 to 0.446, and the EAO on VOT-2017 is increased by 3.5% from 0.248 to 0.283. The third stage produces 0.9% and 0.6% improvements on LaSOT and VOT-2017, respectively. We observe that the improvement by the second stage is higher than that by the third stage. This suggests that most difficult background is handled in the second stage. Adding more stages may lead to further improvements, but also the computation.

Negative anchor filtering? Filtering out the easy negatives aims to provide more balanced training samples for RPN in next stage. To show its effectiveness, we set threshold θ to 1 such that all refined anchors will be send to the next stage. Tab. 5 shows that removing negative anchors in C-RPN can improve the SUC on LaSOT by 1.6% from 0.439 to 0.455, and the EAO on VOT-2017 by 0.7% from 0.282 to 0.289, respectively, which evidences balanced training samples are crucial for training more discriminative RPN.

Feature transfer block? We conduct experiments on two baselines to show the effect of FTB on performance of C-RPN: (a) we only employ the features from one single convolution layer (the last layer) in C-RPN; (b) we utilize multiple layers (the last three layers) in C-RPN but firstly perform correlation for each layer and then fuse the results of all layers (*i.e.*, *decision-level* fusion). Note that, for both baselines (a) and (b), we adopt the cascading strategy. The comparisons of these two baselines with the proposed approach is demonstrated in Tab. 6. We observe that the use of features from multiple layers helps improve the SUC on LaSOT by 0.7% from 0.442 to 0.449, and the EAO on VOT-2017 by 0.4% from 0.278 to 0.282. Further, the combination of FTB and multi-layer features boosts the performance to 0.455 and 0.289 on LaSOT and VOT-2017, respectively, validating the effectiveness of multi-level feature fusion in improving performance.

These studies show that each ingredient brings individual improvement, and all of them work together to produce the excellent tracking performance.

5. Conclusion

In this paper, we propose a novel multi-stage framework C-RPN for tracking. Compared with previous state-of-the-arts, C-RPN demonstrates more robust performance in handling complex background such as similar semantic distractors by performing hard negative sampling within a cascade architecture. In addition, we present a novel FTB module that enables effective feature usage across layers for more discriminative representation. Moreover, C-RPN progressively refines the target bounding box using multiple steps of regressions, leading to more accurate localization. In our extensive experiments on six popular benchmarks, C-RPN consistently achieves the state-of-the-art results and runs in real-time.

Acknowledgement. This work was supported in part by US NSF grants 1814745, 1618398, 1407156, and 1350521.

[3]The result of C-RPN on TrackingNet [34] is evaluated by the server provided by the organizer at http://eval.tracking-net.org/web/challenges/challenge-page/39/leaderboard/42. The results of compared trackers are reported from [34]. Full comparison is shown in the supplementary material.

References

[1] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016. 6, 7, 8

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCVW*, 2016. 1, 3, 4, 6, 7, 8

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 2, 3

[4] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu. Real-time actor-critictracking. In *ECCV*, 2018. 6

[5] Zhizhen Chi, Hongyang Li, Huchuan Lu, and Ming-Hsuan Yang. Dual deep network for visual tracking. *TIP*, 26(4):2005–2015, 2017. 7

[6] Jongwon Choi, Hyung Jin Chang, Tobias Fischer, Sangdoo Yun, Kyuewang Lee, Jiyeoup Jeong, Yiannis Demiris, and Jin Young Choi. Context-aware deep feature compression for high-speed visual tracking. In *CVPR*, 2018. 6, 7

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, Michael Felsberg, et al. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017. 2, 6, 7, 8

[8] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 2, 7

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[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. *arXiv*, 2018. 2, 6, 7, 8

[11] Heng Fan and Haibin Ling. Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In *ICCV*, 2017. 6

[12] Heng Fan and Haibin Ling. Sanet: Structure-aware network for visual tracking. In *CVPRW*, 2017. 2

[13] Hamed Kiani Galoogahi, Ashton Fagg, and Simon Lucey. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017. 6, 7, 8

[14] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *ICCV*, 2015. 2

[15] Erhan Gundogdu and A Aydın Alatan. Good features to correlate for visual tracking. *TIP*, 27(5):2526–2540, 2018. 7

[16] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. Towards a better match in siamese network based visual object tracker. In *ECCVW*, 2018. 3

[17] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018. 1, 3, 6

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

[19] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 1, 3

[20] Matej Kristan et al. The visual object tracking vot2016 challenge results. In *ECCVW*, 2016. 2, 6, 7

[21] Matej Kristan et al. The visual object tracking vot2017 challenge results. In *ICCVW*, 2017. 2, 7, 8

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 4, 6

[23] 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, 3, 4, 5, 6, 7

[24] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 3

[25] Peixia Li, Dong Wang, Lijun Wang, and Huchuan Lu. Deep visual tracking: Review and experimental comparison. *PR*, 76:323–338, 2018. 2

[26] Yang Li and Jianke Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCVW*, 2014. 8

[27] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 2

[28] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[29] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3

[31] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017. 7, 8

[32] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 2, 3, 6

[33] Matthias Mueller, Neil Smith, and Bernard Ghanem. Context-aware correlation filter tracking. In *CVPR*, 2017. 8

[34] 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, 7, 8

[35] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 2, 7, 8

[36] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang. Hedged deep tracking. In *CVPR*, 2016. 6

[37] 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

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 4, 5

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 6

[40] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 2

[41] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *TPAMI*, 36(7):1442–1468, 2014. 2

[42] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *ICCV*, 2017. 6

[43] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *CVPR*, 2018. 2, 7

[44] Chong Sun, Huchuan Lu, and Ming-Hsuan Yang. Learning spatial-aware regressions for visual tracking. In *CVPR*, 2018. 7

[45] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 1, 3, 6

[46] Jack Valmadre, Luca Bertinetto, João Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017. 1, 3, 6, 8

[47] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM MM*, 2015. 6

[48] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2, 3

[49] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 2, 3

[50] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013. 2

[51] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: residual attentional siamese network for high performance online visual tracking. In *CVPR*, 2018. 1, 3

[52] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 2, 6

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

[54] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM CSUR*, 38(4):13, 2006. 2

[55] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018. 3

[56] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *CVPR*, 2017. 7

[57] Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *ECCV*, 2018. 7

[58] Gao Zhu, Fatih Porikli, and Hongdong Li. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *CVPR*, 2016. 7

[59] 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, 3, 6