SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking

Dongyan Guo †, Jun Wang †, Ying Cui †*, Zhenhua Wang †, Shengyong Chen ‡
† Zhejiang University of Technology, China
‡ Tianjin University of Technology, China
{guodongyan, 1111912011, cuiying, zhhwang}@zjut.edu.cn, sy@ieee.org

Abstract

By decomposing the visual tracking task into two subproblems as classification for pixel category and regression for object bounding box at this pixel, we propose a novel fully convolutional Siamese network to solve visual tracking end-to-end in a per-pixel manner. The proposed framework SiamCAR consists of two simple subnetworks: one Siamese subnetwork for feature extraction and one classification-regression subnetwork for bounding box prediction. Different from state-of-the-art trackers like Siamese-RPN, SiamRPN++ and SPM, which are based on region proposal, the proposed framework is both proposal and anchor free. Consequently, we are able to avoid the tricky hyper-parameter tuning of anchors and reduce human intervention. The proposed framework is simple, neat and effective. Extensive experiments and comparisons with state-of-the-art trackers are conducted on challenging benchmarks including GOT-10K, LaSOT, UAV123 and OTB-50. Without bells and whistles, our SiamCAR achieves the leading performance with a considerable real-time speed. The code is available at https://github.com/ohhhyeahhh/SiamCAR.

1. Introduction

Visual object tracking has received considerable attention due to its wide application such as intelligent surveillance, human-machine interaction and unmanned vehicles. Rapid progress has been made on visual tracking. However, it remains a challenging task especially for real world applications, as object in unconstrained recording conditions often suffers from large illumination variation, scale variation, background clutters and heavy occlusions, etc. Moreover, the appearance of non-rigid objects may change significantly due to extreme pose variation.

Current popular visual tracking methods [1, 21, 14, 35, 20, 42, 11, 33] revolve around the Siamese network based architectures. The Siamese network formulates the visual tracking task as a target matching problem and aims to learn a general similarity map between the target template and the search region. Since one single similarity map typically contains limited spatial information, a common strategy is to perform matching on multiple scales of the search regions to determine the object scale variation [1, 14, 35], which explains why these trackers are time-consuming and labor-intensive. SiamRPN [21] attaches the Siamese network a subnetwork for the extraction of region proposals (RPN). By jointly training a classification branch and a regression branch for visual tracking, SiamRPN avoids the time-consuming step of extracting multi-scale feature maps for the object scale invariance. It achieves state-of-the-art results on multiple benchmarks. Later works such as DaSiam [42] , CSiam [11] and SiamRPN++ [20] improve SiamRPN. However, since anchors are introduced for region proposal, these trackers are sensitive to the numbers, sizes and aspect ratios of anchor boxes, and expertise on hyper-parameter tuning is crucial to obtain successful tracking with these trackers.

Figure 1. Comparisons of the proposed SiamCAR with three state-of-the-art trackers on three challenging sequences from GOT-10K. Our SiamCAR can accurately predict the bounding boxes even when objects suffer from similar distractors, large scale variation and large pose variation, while SiamRPN++ and SPM give much rougher results and ECO drifts to the background.

In this paper, we show that an anchor-free Siamese network based tracker can perform better than the state-of-the-art RPN based trackers. Essentially we decompose tracking into two subproblems: one classification problem and one regression task. The classification branch aims to predict each spatial location a label, while the regression branch considers regressing each location one relative bounding box. With such decomposition, the tracking task can be solved in a per-pixel prediction manner. We then craft a simple yet effective Siamese based classification and regression network (SiamCAR) to learn the classification and regression models simultaneously in an end-to-end manner.

Previous work [4] leverages object semantic information to improve the bounding box regression. Inspired by this, SiamCAR is designed to extract response maps which include affluent category information and semantic information. Different from RPN models [21, 42, 20], which use two response maps for region proposal detection and regression respectively, SiamCAR takes one unique response map to predict object location and bounding box directly.

SiamCAR adopts the strategy of online training and offline tracking, without using any data enhancement during training. Our main contributions are:

- We propose the so-called Siamese classification and regression framework (SiamCAR) for visual tracking. The framework is very simple in construction but powerful in performance.
- The proposed tracker is both anchor and proposal free. The number of hyper-parameters has been significantly reduced, which keeps the tracker from complicated parameter tuning during training.
- Without bells and whistles, the proposed tracker achieves the state-of-the-art tracking performance in terms of both accuracy and speed.

2. Related Work

We mainly review the family of Siamese RPN trackers since they dominate the tracking performance in recent years.

Tracking researchers devote to design faster and more accurate trackers from different aspects like feature extraction [16, 17], template updating [35, 12], classifier design [40] and bounding box regression [4]. Early feature extraction mainly uses color features, texture features or other hand-crafted ones. Due to the rapid progress of deep learning, convolutional neural network (CNN)-based feature has widely been adopted. Though the adaptability of trackers could be improved via template updating, the online tracking is less efficient. Besides, template updating suffers from tracking drift. The introduction of correlation filter methods [2, 6, 16, 22, 41, 25] improves the tracking performance significantly in terms of both efficiency and accuracy. Current researches demonstrated that the Siamese based online training and offline tracking approaches with CNNs have achieved the best balance between accuracy and efficiency [21, 20].

As one of the pioneering works, SiamFC [1] constructs a fully convolutional Siamese network to train a tracker. Encouraged by its success, many researchers follow the work and propose some updated models [9, 35, 14, 13, 21, 20] . CFNet [35] introduces the Correlation Filter layer to the SiamFC framework and performs online tracking to improve the accuracy. By modifying the Siamese branches with two online transformations, DSiam [13] proposes to learn a dynamic Siamese network, which achieves better tracking accuracy with acceptable tradeoff of speed. SA-siam [14] builds a two-fold Siamese network with a semantic branch and an appearance branch. The two branches are trained separately to keep the heterogeneity of features but combined at the testing time to improve the tracking accuracy. In order to tackle the scale variation problem, these Siamese networks resort to multi-scale searching and result in increase of time-consuming.

Inspired by the region proposal network for object detection [31], the SiamRPN [21] tracker performs the region proposal extraction using the output of Siamese network. By jointly learning a classification branch and a regression branch for region proposal, SiamRPN avoids the time-consuming step of extracting multi-scale feature maps. However, it has difficulty in dealing with distractors with similar appearance to the target object. Based on SiamRPN, DaSiamRPN [42] increases the hard negative training data during the training phase. Through data enhancement, they improve the discrimination of the tracker, hence a much more robust tracking result is obtained. The tracker is further extended to long-term visual tracking. Though these aforementioned approaches modified the original SiamFC [1] on many aspects, the performance stall mainly because the backbone network they used (AlexNet) is weak. SiamRPN++ [20] replaces AlexNet with ResNet [15]. Meanwhile it randomly shifts the training object location in the search region during training to eliminate the center bias. Such modifications boost tracking accuracy.

Anchors are adopted in these RPN based trackers for region proposal. Besides, anchor boxes can make use of the deep feature maps and avoid repeated computation, which can significantly speed up the tracking process. The state-of-the-art trackers, such as SPM [36] and SiamRPN [21] track in a very high frame rate. Though SiamRPN++ [20] adopts a very deep neural network, it still works in a considerable real-time speed. The accuracy and speed of the state-of-the-art anchor-free trackers (*e.g.*, ECO [3]) are still much worse than these anchor-based trackers [36, 20] on the

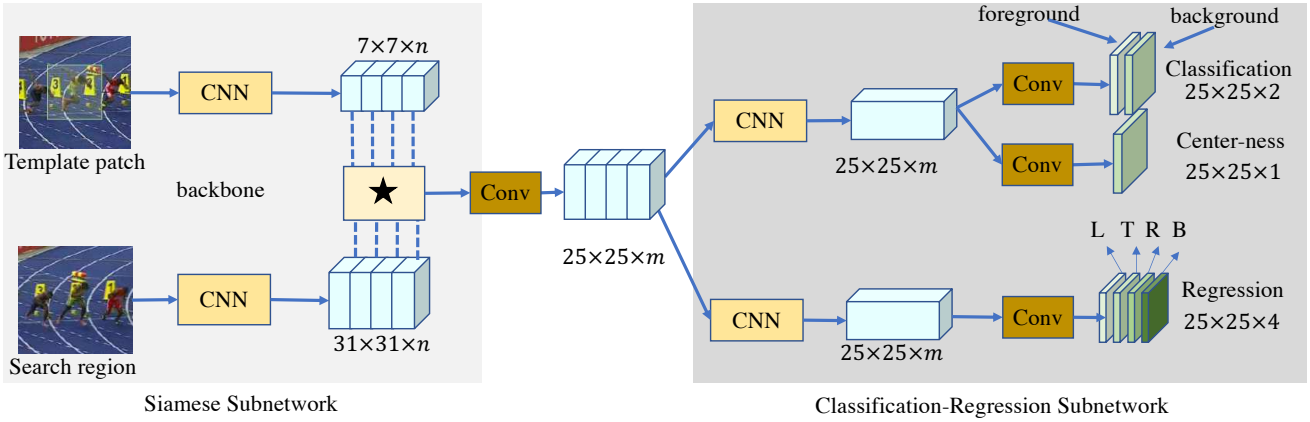

Figure 2. Illustration of SiamCAR: The left side is a Siamese subnetwork with a depth-wise cross correlation layer (denoted by ★) for multi-channel response map extraction. The right side shows the classification and regression subnetwork for bounding box prediction, which is taken to decode the location and scale information of the object from multi-channel response map. Note that SiamCAR can be implemented as a fully convolutional network, which is simple, neat and easy to interpret.

challenging benchmarks like GOT-10K [18]. Nevertheless, the performance of anchor-based trackers is very sensitive to the hyper-parameters of anchors, which need to be carefully tuned to achieve ideal performance. Moreover, since the size and aspect ratio of anchor boxes are fixed, even with tuned parameters, these trackers still have difficulty in processing objects with large shape deformation and pose variation. In this paper, we show that these issues could be significantly alleviated with our proposed SiamCAR. Meanwhile, we demonstrate that a tracker with much simpler architecture can achieve even better performance than the state-of-the-arts.

3. Proposed Method

We now introduce our SiamCAR network in detail. As mentioned, we decompose the tracking task into two subproblems as classification and regression, and then solve them in a per-pixel manner. As show in Figure 2, the framework mainly consists of two simple subnetworks: one Siamese network for feature extraction and the other network for bounding box prediction.

3.1. Feature Extraction

Here we take advantage of the fully convolution network to construct the Siamese subnetwork for the visual feature extraction. The subnetwork consists of two branches: a target branch which takes the tracking template patch Z as input, and a search branch which takes the search region X as input. The two branches share the same CNN architecture as their backbone models, which output two feature maps $\varphi(Z)$ and $\varphi(X)$. In order to embed the information of these two branches, a response map R can be obtained by performing the cross-correlation on $\varphi(X)$ with $\varphi(Z)$ as a kernel. Since we need to decode the response map R in the subsequent prediction subnetwork to obtain the location and scale information of the target, we hope that R retains abundant information. However, the cross-correlation layer can only generate a single-channel compressed response map, which lacks useful features and important information for tracking, as suggested by [20] that different feature channels typically take distinct semantic information. Inspired by [20], we also use a depth-wise correlation layer to produce multiple semantic similarity maps:

$$R = \varphi(X) \star \varphi(Z), \tag{1}$$

where $\star$ denotes the channel-by-channel correlation operation. The generated response map R has the same number of channels as $\varphi(X)$, and it contains massive information for classification and regression.

Low-level features like edge, corner, color and shape that represent better visual attributes are indispensable for location, while high-level features are better to represent semantic attributes that are crucial for discrimination. Many methods take advantage of fusing both low-level and high-level features to improve the tracking accuracy [27, 20]. Here we also consider to aggregate multi-layer deep features for tracking. We use the modified ResNet-50 as the same in [20] as our backbone networks. To achieve better inference for recognition and discrimination, we combine the features extracted from the last three residual blocks of the backbone, which are denoted respectively as $\mathcal{F}_3(X)$, $\mathcal{F}_4(X)$, $\mathcal{F}_5(X)$. Specifically, we perform a channel-wise concatenation:

$$\varphi(X) = Cat(\mathcal{F}_3(X), \mathcal{F}_4(X), \mathcal{F}_5(X)), \tag{2}$$

where $\mathcal{F}_{i=3:5}(X)$ includes 256 channels. Consequently $\varphi(X)$ contains 3×256 channels.

The Depth-wise Cross Correlation is performed between the searching map $\varphi(X)$ and the template map $\varphi(Z)$ to get a multi-channel response map. The response map is then convoluted with a 1×1 kernel to reduce its dimension to 256 channels. Through the dimension-reduction, the number of parameters can be significantly reduced, and the following computation can be sped up. The final dimension-reduced response map R^* is adopted as the input to the classification-regression subnetwork.

3.2. Bounding Box Prediction

Each location (i, j) in the response map R^* can be mapped back onto the input search region as (x, y). The RPN-based trackers consider the corresponding location on the search region as the center of multi-scale anchor boxes, and regress the target bounding box with these anchor boxes as references. Different from them, our network directly classifies and regresses the target bounding box at each location. The associated training can be accomplished by the fully convolution operation in an end-to-end fashion, which avoids tricky parameter tuning and reduces human intervention.

The tracking task is decomposed into two subtasks: a classification branch to predict the category for each location, and a regression branch to compute the target bounding box at this location (see Figure 2 for an illustration of the subnetwork). For a response map $R^*_{w\times h\times m}$ extracted using the Siamese subnetwork, the classification branch outputs a classification feature map $A^{cls}_{w\times h\times 2}$ and the regression branch outputs a regression feature map $A^{reg}_{w\times h\times 4}$. Here w and h represent the width and the height of the extracted feature maps respectively. As that shown in Figure 2, each point $(i, j, :)$ in $A^{cls}_{w\times h\times 2}$ contains a 2D vector, which represents the foreground and background scores of the corresponding location in the input search region. Similarly, each point $(i, j, :)$ in $A^{reg}_{w\times h\times 4}$ contains a 4D vector $t(i, j) = (l, t, r, b)$, which represents the distances from the corresponding location to the four sides of the bounding box in the input search region.

Since the ratio of areas occupied by the target and the background in the input search region is not very large, sample imbalance is not a problem. Therefore, we simply adopt the cross-entropy loss for classification and the IOU loss for regression. Let (x_0, y_0) and (x_1, y_1) denote the left-top and right-bottom corner of the ground truth bounding box, and let (x, y) denote the corresponding location of point (i, j), the regression targets $\tilde{t}_{(i,j)}$ at $A^{reg}_{w\times h\times 4}(i, j, :)$ can be calculated by:

$$\begin{aligned} \tilde{t}^0_{(i,j)} &= \tilde{l} = x - x_0, \tilde{t}^1_{(i,j)} = \tilde{t} = y - y_0, \\ \tilde{t}^2_{(i,j)} &= \tilde{r} = x_1 - x, \tilde{t}^3_{(i,j)} = \tilde{b} = y_1 - y. \end{aligned} \tag{3}$$

With $\tilde{t}_{(i,j)}$, the IOU between the ground-truth bounding box and the predicted bounding box can be computed. Then we compute the regression loss using

$$\mathcal{L}_{reg} = \frac{1}{\sum \mathbb{I}(\tilde{t}_{(i,j)})} \sum\nolimits_{i,j} \mathbb{I}(\tilde{t}_{(i,j)}) L_{IOU}(A^{reg}(i, j, :), \tilde{t}_{(x,y)}), \tag{4}$$

where L_{IOU} is the IOU loss as in [38] and $\mathbb{I}(\cdot)$ is an indicator function defined by:

$$\mathbb{I}(\tilde{t}_{(i,j)}) = \begin{cases} 1 & \text{if } \tilde{t}^k_{(i,j)} > 0, k = 0, 1, 2, 3 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

An observation is that the locations far away from the center of an target tend to produce low-quality predicted bounding boxes, which reduces the performance of the tracking system. Following [34], we add a centerness branch in parallel with the classification branch to remove the outliers. As shown in Figure 2, the branch outputs a centerness feature map $A^{cen}_{w\times h\times 1}$, where each point value gives the centerness score of the corresponding location. The score $C(i, j)$ in $A^{cen}_{w\times h\times 1}(i, j)$ is defined by

$$C(i, j) = \mathbb{I}(\tilde{t}_{(i,j)}) * \sqrt{\frac{min(\tilde{l}, \tilde{r})}{max(\tilde{l}, \tilde{r})} \times \frac{min(\tilde{t}, \tilde{b})}{max(\tilde{t}, \tilde{b})}}, \tag{6}$$

where $C(i, j)$ is in contrast with the distance between the corresponding location (x, y) and the object center in the search region. If (x, y) is a location within background, the value of $C(i, j)$ is set to 0. The centerness loss is

$$\begin{aligned} \mathcal{L}_{cen} = \frac{-1}{\sum \mathbb{I}(\tilde{t}_{(i,j)})} \sum\nolimits_{\mathbb{I}(\tilde{t}_{(i,j)})==1} & C(i, j) * \log A^{cen}_{w\times h\times 1}(i, j) \\ & + (1 - C(i, j)) * \log(1 - A^{cen}_{w\times h\times 1}(i, j)). \end{aligned} \tag{7}$$

The overall loss function is

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda_1 \mathcal{L}_{cen} + \lambda_2 \mathcal{L}_{reg}, \tag{8}$$

where $\mathcal{L}_{cls}$ represents the cross-entropy loss for classification. Constants λ_1 and λ_2 weight the centerness loss and the regression loss. During training, we empirically set $\lambda_1 = 1$ and $\lambda_2 = 3$ for all experiments.

3.3. The Tracking Phase

Tracking aims at predicting a bounding box for the target in current frame. For a location (i, j), the proposed method produces a 6D vector $T_{ij} = (cls, cen, l, t, r, b)$, where cls represents the foreground score of classification, cen represents the centerness socre, and $l + r$ and $t + b$ represent the predicted width and height of the target in current frame. During tracking, the size and aspect ratio of the bounding box typically see minor change across consecutive frames. To supervise the prediction using this spatial-temporal consistency, we adopt a scale

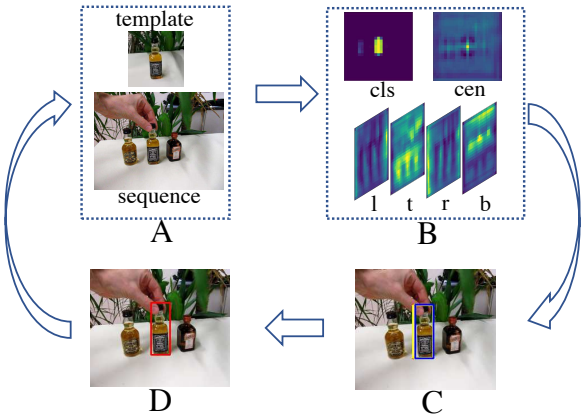

Figure 3. Tracking process: Sub-figure A shows a pair of inputs while B presents the corresponding outputs of the model, where we show our model gives good prediction for different attributes of the object. C shows the predicted bounding boxes corresponding to the top-k points. D shows the final predicted bounding box by averaging those boxes in C.

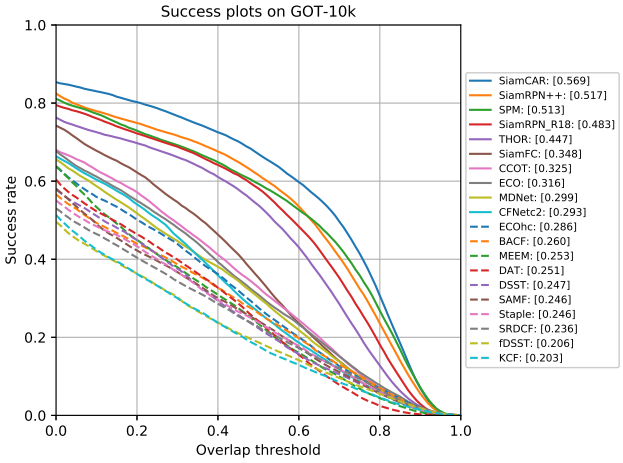

Figure 4. Comparisons on GOT-10K [18]. Our SiamCAR significantly outperforms the baselines and other state-of-the-art methods.

change penalty p_{ij} as that introduced in [21] to re-rank the classification score cls, which admits an updated 6D vector $PT_{ij} = (cls_{ij} \times p_{ij}, cen, l, t, r, b)$. Then the tracking phase can be formulated as:

$$q = \arg\max_{i,j}\{(1-\lambda_d)cls_{ij} \times p_{ij} + \lambda_d H_{ij}\}, \quad (9)$$

where H is the cosine window and λ_d is the balance weight. The output q is a queried location with the highest score being a target pixel.

Since our model solves the object tracking with a per-pixel prediction manner, each location is relative to a predicted bounding box. In the real tracking process, it will be jittering between adjacent frames if only one bounding box of q is used as the target box. We observed that the pixels located around q are more likely to be the target pixel. Hence we choose the top-k points from n neighborhoods of q according to the value $cls_{ij} \times p_{ij}$. The final prediction is the weighted average of the selected k regression boxes. Empirically, we found that setting $n = 8$ and $k = 3$ delivers stable tracking results.

4. Experiments

4.1. Implementation Details

The proposed SiamCAR is implemented in Python with PyTorch and trained on 4 RTX 2080 Ti cards. For fair comparison, the input size of the template patch and search regions are set as the same with [20], respectively to 127 pixels and 255 pixels. The modified ResNet-50 as in [20] is adopted as the backbone of our Siamese subnetwork. The network is pretrained on ImageNet [32]. Then we use the pretrained weights as initialization to train our model.

Training Details. During the training process, the batch size is set as 96 and totally 20 epochs are performed by using stochastic gradient descent (SGD) with an initial learning rate 0.001. For the first 10 epochs, the parameters of the Siamese subnetwork are frozen when training the classification and regression subnetwork. For the last 10 epochs, the last 3 blocks of ResNet-50 are unfrozen for training. The whole training phase takes around 42 hours. We train our SiamCAR with the data from COCO [24], ImageNet DET, ImageNet VID [32] and YouTube-BB [30] for experiments on UAV and OTB [37]. It should be noticed that for experiments on GOT-10K [18] and LaSOT [10], our SiamCAR is trained with only the specified training set provided by the official website for fair comparison.

Testing Details. The testing phase uses the offline tracking strategy. Only the object in first frame of a sequence is adopted as the template patch. Consequently, the target branch of the Siamese subnetwork can be pre-computed and fixed during the whole tracking period. The search region in the current frame is adopted as the input of the search branch. In Figure 3 we show the whole tracking process. With the outputs of classification-regression subnetwork, a location q is queried through Equation (9). In order to achieve a more stable and smoother prediction between adjacent frames, a weighted average of regression boxes corresponding to the top-3 neighbors of q is computed as the final tracking result. For evaluations on different datasets, we use the official measurements provided there, which can be different from each other.

4.2. Results on GOT-10K

GOT-10K [18] is a recently released large-scale and high-diversity benchmark for generic object tracking in the wild. It contains more than 10,000 video segments of real-

Tracker	AO	$SR_{0.5}$	$SR_{0.75}$	FPS	$Hardware$	$Language$
KCF [16]	0.203	0.177	0.065	94.66	CPU	Matlab
fDSST [6]	0.206	0.187	0.075	30.43	CPU	Matlab
SRDCF [5]	0.236	0.227	0.094	5.58	CPU	Matlab
Staple [26]	0.246	0.239	0.089	28.87	CPU	Matlab
SAMF [23]	0.246	0.241	0.084	7.43	CPU	Matlab
DSST [7]	0.247	0.223	0.081	18.25	CPU	Matlab
DAT [29]	0.251	0.242	0.048	45.52	CPU	Matlab
MEEM [39]	0.253	0.235	0.068	20.59	CPU	Matlab
BACF [19]	0.260	0.262	0.101	14.44	CPU	Matlab
ECO-HC	0.286	0.276	0.096	44.55	CPU	Matlab
CFnet [35]	0.293	0.265	0.087	35.62	Titan X	Matlab
MDnet [28]	0.299	0.303	0.099	1.52	Titan X	Python
ECO [3]	0.316	0.309	0.111	2.62	CPU	Matlab
CCOT [8]	0.325	0.328	0.107	0.68	CPU	Matlab
SiamFC [1]	0.374	0.404	0.144	25.81	Titan X	Matlab
THOR	0.447	0.538	0.204	1.00	RTX 2070	Python
SiamRPN_R18	0.483	0.581	0.270	97.55	Titan X	Python
SPM [36]	0.513	0.593	**0.359**	72.30	Titan Xp	Python
SiamRPN++ [20]	**0.517**	**0.616**	0.325	49.83	RTX 2080ti	Python
SiamCAR	**0.569**	**0.670**	**0.415**	52.27	RTX 2080ti	Python

Table 1. The evaluation on GOT-10K [18]. Top-2 results are highlighted in red and blue respectively.

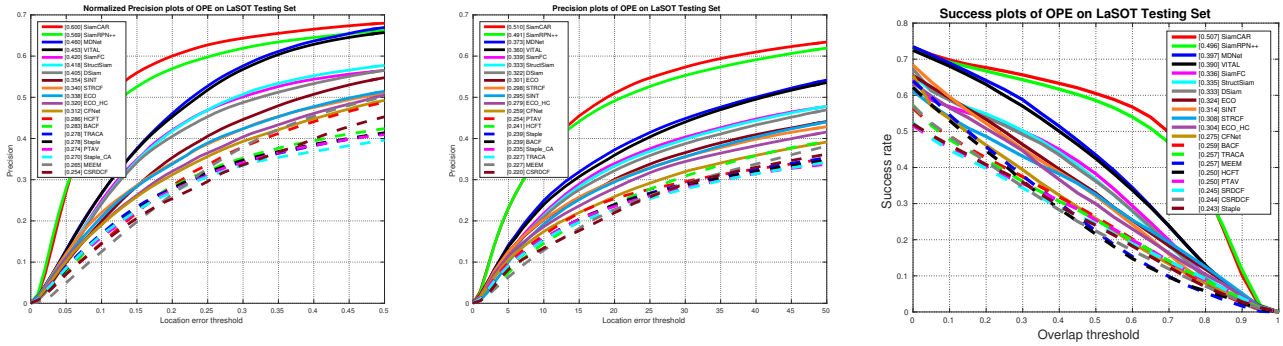

Figure 5. Comparison with top-20 trackers on LaSOT [10]. Our SiamCAR significantly outperforms baselines and the state-of-the-arts.

world moving objects. Fair comparison of deep trackers is ensured with the protocol that all approaches use the same training and testing data provided by the dataset. The classes in training dataset and testing dataset are zero overlapped. After uploading the tracking results, the analysis is taken automatically by the official website. The provided evaluation metrics include success plots, average overlap (AO) and success rate (SR). The AO represents the average overlaps between all estimated bounding boxes and ground-truth boxes. The $SR_{0.5}$ represents the rate of successfully tracked frames whose overlap exceeds 0.5, while $SR_{0.75}$ represents this overlap exceeds 0.75.

We evaluate SiamCAR on GOT-10K and compare it with state-of-the-art trackers including SiamRPN++ [20], SiamRPN [21], SiamFC [1], ECO [3], CFNET [35] and other baselines or state-of-the art approaches. All the results are provided by the official website of GOT-10K. Figure 1 shows that SiamCAR can outperforms all the trackers on GOT-10K and Table 1 lists quantitative results on different metrics. Clearly, our tracker performs best in terms of all metrics. Compared with SiamRPN++, SiamCAR improves the scores by 5.2%, 5.4% and 9.0% respectively for AO, $SR_{0.5}$ and $SR_{0.75}$.

In Table 1, we also show the tracking frame rate in frame-per-second (FPS). The reported speed is evaluated on a machine with one RTX2080ti and others are provided by the GOT-10K official results. As that shown here, our SiamCAR is much faster than most evaluated trackers at a real-time speed of 52.27 FPS.

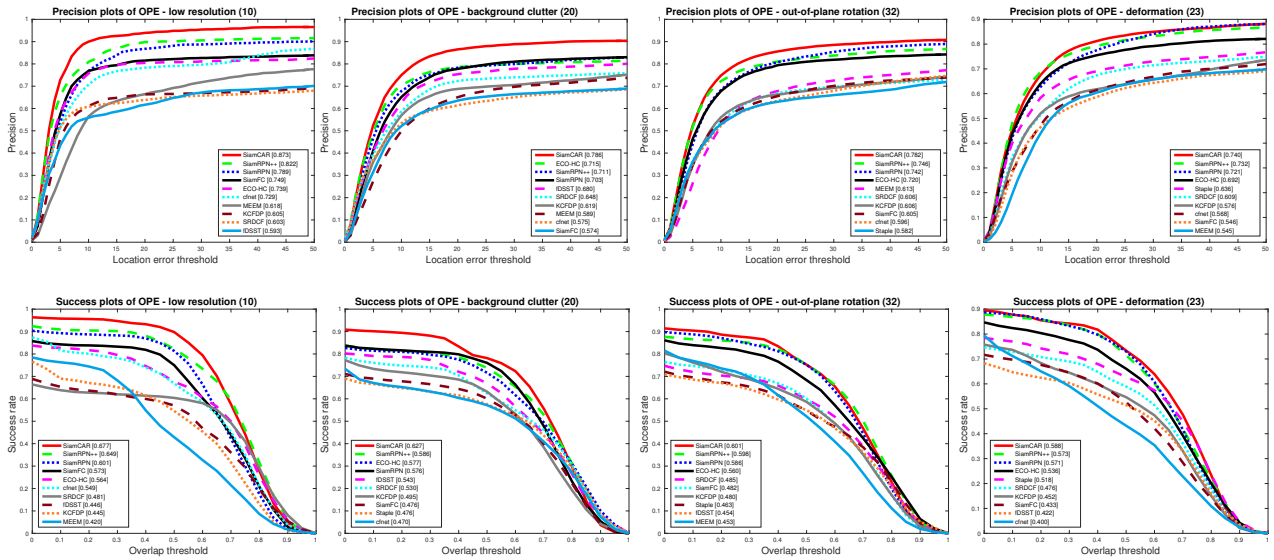

Figure 6. The evaluation on OTB-50 [37] with challenging aspects including low resolution, background clutter, out-of-plane rotation and deformation. Our SiamCAR achieves the best results against the impacts of all these aspects.

4.3. Results on LaSOT

LaSOT is a recently released benchmark for single object tracking. The dataset contains more than 3.52 million manually annotated frames and 1400 videos. It contains 70 classes and each class includes 20 tracking sequences. Such a large-scale dataset brings a great challenge to tracking algorithms. The official website of LaSOT provides 35 algorithms as baselines. Normalized precision plots, precision plots and success plots in one-pass evaluation (OPE) are considered as the evaluation metrics.

We compare our SiamCAR with the top-19 trackers including SiamRPN++ [20], MDNet [28], DSiam [13], ECO [3] and other baselines. The results of SiamRPN++ [20] are provided by the website of its authors, while other results are provided by the official website of LaSOT. As shown in Figure 5, our SiamCAR achieves the best performance. Compared with SiamRPN++, our SiamCAR improves the scores by 3.1%, 1.9% and 1.1% respectively for the three metrics. Notably, our SiamCAR improves by over 14%, 13.7% and 11% respectively for the three indicators in comparison with the baseline approaches.

The leading results on such a large dataset demonstrate that our proposed network has a good generalization for visual object.

4.4. Results on OTB50

OTB-50 contains 50 challenging videos with substantial variations. The test sequences are manually tagged with 9 attributes to represent the challenging aspects, including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters and low resolution. We compare our network with 9 state-of-the-art approaches including SiamRPN++ [20], SiamRPN [21], SiamFC [1] and ECO [3]. We evaluate success plots and precision plots in OPE for each tracker. As shown in Figure 6, the proposed SiamCAR ranks the first in terms of both metrics. Especially, our SiamCAR significantly improves the tracking accuracy against the impacts of low resolution, out-of-plane rotation and background clutter. The results demonstrate that SiamCAR can better deal with challenging distractors and large pose variation, which benefits from the implicitly decoded semantic information by our classification-regression subnetwork.

backbone	ResNet-50	ResNet-34	alexnet
Precision	0.76	0.751	0.713
FPS	52	77	170

Table 2. Comparisons on UAV123 with different backbone architecture.

4.5. Results on UAV123

UAV123 dataset contains 123 video sequences and more than $110K$ frames. All sequences are fully annotated with upright bounding boxes. Objects in the dataset see fast motion, large scale and illumination variations and occlusions, which make tracking using this dataset challenging.

We compare our SiamCAR with 9 state-of-the-art approaches including SiamRPN++ [20], SiamRPN [21], SiamFC [1] and ECO [3] on this dataset. The success plot and precision plot of OPE are used to evaluate the overall

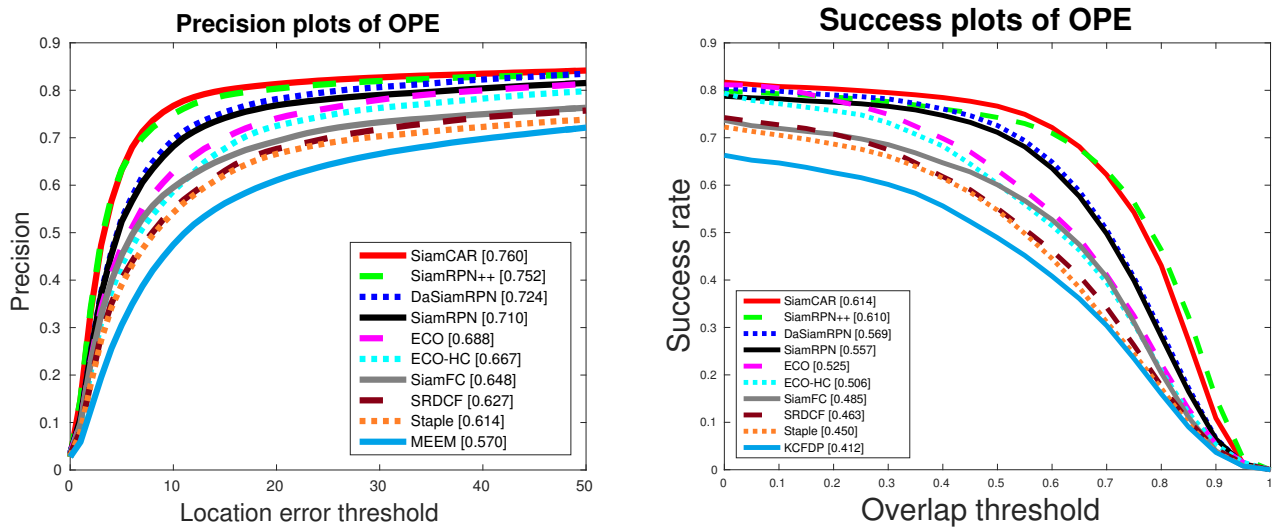

Figure 7. The evaluation on UAV123. Our SiamCAR performs the best on all evaluation metrics.

performance here. As shown in Figure 7, our SiamCAR outperforms all other trackers on both metrics. Compared with state-of-the-art RPN trackers [20, 42, 21], SiamCAR obtains competitive results with a much simpler network, and it does not require to tune parameters heuristically.

4.6. Backbone Architecture Evaluation

To verify the effectiveness of the proposed framework, we compare different backbone architectures for object tracking. Table 2 shows the tracking performance using ResNet-50, ResNet-34 and AlexNet as backbones. We report results with respect to precision and frame-per-second (FPS) on UAV123 by replacing the backbone. One can see that the proposed SiamCAR can achieve comparable results with different backbones. Notably, a speed of 170 FPS can be achieved with Alexnet. Obviously the proposed framework can benefit from deeper networks. By replacing AlexNet with ResNet50, the precision increase by around 6.5% while the tracking speed decreases to 52 FPS, which is still in real-time speed. The evaluation also suggests that by changing the backbone network, it is easy to fit the proposed SiamCAR to different real tasks with a trade-off between accuracy and efficiency.

5. Conclusions

In this paper, we have presented a Siamese classification and regression framework, namely SiamCAR which enables the end-to-end training of a deep Siamese network for visual tracking. We show that tracking tasks can be resolved in a per-pixel manner using the proposed neat fully convolution framework. The proposed framework is very simple in terms of its architecture but achieves new state-of-the-art results without bells and whistles on GOT-10K and other challenging benchmarks. It also achieves the best performance on large-scale dataset like LaSOT, which verifies the generalization ability of the proposed framework. Since our SiamCAR is simple and neat, several modifications could be performed next to achieve further improvement.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (2018YFB1305200), the National Natural Science Foundation of China (61802348), and in part by Natural Science Foundation of Zhejiang Province (LQ18F030013, LQ18F030014).

References

[1] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, and P.H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

[2] D. Bolme, J. Beveridge, B. Draper, and Y. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[3] M. Danelljan, G. Bhat, F.S. Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

[4] M. Danelljan, G. Bhat, F.S. Khan, and M. Felsberg. Accurate tracking by overlap maximization. In *CVPR*, 2019.

[5] M. Danelljan, G. Hager, K.S. Fahad, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015.

[6] M. Danelljan, G. Hager, K.S. Fahad, and M. Felsberg. Discriminative scale space tracking. *TPAMI*, 2016.

[7] M. Danelljan, G. Hager, and F. Khan. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[8] M. Danelljan, A. Robinson, F.S. Khan, and M. Felsberg. Beyond correlation filters:learning continuous convolution operators for visual tracking. In *ECCV*, 2016.

[9] X.P. Dong and J.B. Shen. Triplet loss in siamese network for object tracking. In *ECCV*, 2018.

[10] H. Fan, L.T. Lin, F. Yang, P. Chu, G. Deng, S.J. Yu, H.X. Bai, Y. Xu, C.Y. Liao, and H.B. Ling. Lasot: A high-quality benchmark for large-scale single object tracking. In *CVPR*, 2019.

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

[12] J.Y. Gao, T.Z. Zhang, and C.S. Xu. Graph convolutional tracking. In *CVPR*, 2019.

[13] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, 2017.

[14] A.F. He, C. Luo, X.M. Tian, and W.J. Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] Joao.F. Henriques, R. Caseiro, M. Pedro, and B. Jorge. High-speed tracking with kernelized correlation filters. *TPAMI*, 2014.

[17] P. Horst, M. Thomas, and B. Horst. In defense of color-based model-free tracking. In *CVPR*, 2015.

[18] L.H. Huang, X. Zhao, and K.Q. Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *TPAMI*, 2018.

[19] G. Kiani, Hamed, F. Ashton, and L. Simon. Learning background-aware correlation filters for visual tracking. In *ICCV*, 2017.

[20] B. Li, W. Wu, Q. Wang, F.Y. Zhang, J.L. Xing, and J.J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019.

[21] B. Li, J.J. Yan, W. Wu, Z. Zhu, and X.L. Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[22] F. Li, Y.J. Yao, P.H. Li, D. Zhang, W.M. Zuo, and M.H. Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ICCV*, 2017.

[23] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *ECCV*, 2014.

[24] T.Y. Lin, M. Michael, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C.L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[25] T. Liu, G. Wang, Q.X. Yang, and L. Wang. Part-based tracking via discriminative correlation filters. *TCSVT*, 2016.

[26] B. Luca, V. Jack, G. Stuart, M. Ondrej, and T. Philip HS. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[27] C. Ma, J.B. Huang, X.K. Yang, and M.H. Yang. Robust visual tracking via hierarchical convolutional features. *TPAMI*, 2018.

[28] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[29] S. Pu, Y. Song, and C. Ma. Deep attentive tracking via reciprocative learning. In *NIPS*, 2018.

[30] 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. In *CVPR*, 2017.

[31] S.Q. Ren, K.M. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[32] 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. *IJCV*, 2015.

[33] Y.B. Song, C. Ma, L.J. Gong, J.W. Zhang, R.W. Lau, and M.H. Yang. Crest: Convolutional residual learning for visual tracking. In *ICCV*, 2017.

[34] Z. Tian, C.H. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. *ICCV*, 2019.

[35] J. Valmadre, L. Bertinetto, J.F. Henriques, A. Vedaldi, and P.H. Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[36] G.T. Wang, C. Luo, Z.W. Xiong, and W.J. Zeng. Spm-tracker: Series-parallel matching for real-time visual object tracking. 2019.

[37] Y. Wu, J. Lim, and M.H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[38] J.H. Yu, Y.N. Jiang, Z.Y. Wang, Z.M. Cao, and T. Huang. Unitbox: An advanced object detection network. In *MM*, 2016.

[39] J.M. Zhang, S.G. Ma, and S. Stan. Meem: robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014.

[40] L. Zhang, V. Jagannadan, N.S. Ponnuthurai, A. Narendra, and M. Pierre. Robust visual tracking using oblique random forests. In *CVPR*, 2017.

[41] T.Z. Zhang, C.S. Xu, and M.H. Yang. Multi-task correlation particle filter for robust object tracking. In *CVPR*, 2017.

[42] Z. Zhu, Q. Wang, B. Li, W. Wu, J.J. Yan, and W.M. Hu. Distractor-aware siamese networks for visual object tracking. In *ECCV*, 2018.