



Joint Classification and Regression for Visual Tracking with Fully Convolutional Siamese Networks

Ying Cui¹ · Dongyan Guo¹ · Yanyan Shao¹ · Zhenhua Wang¹ · Chunhua Shen² · Liyan Zhang³ · Shengyong Chen⁴

Received: 11 April 2021 / Accepted: 30 November 2021 / Published online: 6 January 2022
© The Author(s) 2022

Abstract

Visual tracking of generic objects is one of the fundamental but challenging problems in computer vision. Here, we propose a novel fully convolutional Siamese network to solve visual tracking by directly predicting the target bounding box in an end-to-end manner. We first reformulate the visual tracking task as two subproblems: a classification problem for pixel category prediction and a regression task for object status estimation at this pixel. With this decomposition, we design a simple yet effective Siamese architecture based classification and regression framework, termed SiamCAR, which consists of two subnetworks: a Siamese subnetwork for feature extraction and a classification-regression subnetwork for direct bounding box prediction. Since the proposed framework is both proposal- and anchor-free, SiamCAR can avoid the tedious hyperparameter tuning of anchors, considerably simplifying the training. To demonstrate that a much simpler tracking framework can achieve superior tracking results, we conduct extensive experiments and comparisons with state-of-the-art trackers on a few challenging benchmarks. Without bells and whistles, SiamCAR achieves leading performance with a real-time speed. Furthermore, the ablation study validates that the proposed framework is effective with various backbone networks, and can benefit from deeper networks. Code is available at <https://github.com/ohhhyeahhh/SiamCAR>.

Keywords Fully convolutional networks · Anchor-free object detection · Visual tracking

Communicated by Gang Hua.

✉ Dongyan Guo
guodongyan@zjut.edu.cn

Ying Cui
cuiying@zjut.edu.cn

Yanyan Shao
2111912036@zjut.edu.cn

Zhenhua Wang
zhhwang@zjut.edu.cn

Chunhua Shen
chunhua@me.com

Liyan Zhang
zhangliyan@nuaa.edu.cn

Shengyong Chen
sy@ieee.org

¹ College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China

² Zhejiang University, Hangzhou, China

1 Introduction

Visual tracking, in particular single object tracking (Ross et al., 2008; Huang et al., 2016; Lukežič et al., 2018; Ma et al., 2018b; Sui et al., 2019), has received considerable attention in the computer vision community due to its wide applications such as intelligent video analysis, human–machine interaction, and autonomous driving. In the past decade, significant progress has been made on visual tracking. However, it remains a challenging task especially for real-world applications, as objects in unconstrained conditions often suffer from illumination/scale variations, background clutters and heavy occlusions. Moreover, the appearance of non-rigid objects may change significantly due to extreme pose variation.

³ College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

⁴ School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China

Researchers endeavor to address challenging issues from different aspects including feature extraction (Henriques et al., 2014; Horst et al., 2015), template updating (Valmadre et al., 2017; Gao et al., 2019), classifier design (Zhang et al., 2017a) and bounding box regression (Danelljan et al., 2019). The introduction of correlation filter methods (Bolme et al., 2010; Danelljan et al., 2016a; Henriques et al., 2014; Li et al., 2017; Zhang et al., 2017b; Liu et al., 2016) improves the tracking performance significantly in terms of both efficiency and accuracy. Early work on feature extraction mainly uses color features, texture features or other hand-crafted ones. Benefiting from the rapid development of deep learning, deep convolutional neural network (CNN)-based features have widely been adopted in recent years (Nam et al., 2016; Ren et al., 2017; Ma et al., 2018a). For general object tracking, the target is only depicted in the first frame of a sequence, hence constructing an offline model yields insufficient target-specific information. To solve the problem, online tracking methods that update model at test-time have shown benefits on adaptiveness (Nam et al., 2016; Bhat et al., 2019a; Danelljan et al., 2019; Bhat et al., 2019b). Although the adaptability of trackers can be improved via template updating, online tracking sometime is less efficient. Besides, template updating often suffers from tracking drift when the target appearance drastically changes across time. Recent studies demonstrate that the Siamese architecture based online training and offline tracking approaches with CNNs have achieved a good balance between accuracy and efficiency (Bertinetto et al., 2016; Li et al., 2018, 2019; Zhu et al., 2018).

Classical Siamese networks such as SiameseFC (Bertinetto et al., 2016) formulate the visual tracking task as a target matching problem and aim 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 (Bertinetto et al., 2016; He et al., 2018; Valmadre et al., 2017), which may explain why these trackers are computationally heavy. SiamRPN (Li et al., 2018) attaches the Siamese network a subnetwork for extraction of region proposals (RPN). Benefiting from the proposal refinement, SiamRPN avoids the time-consuming step of extracting multi-scale feature maps for achieving object scale invariance. It shows state-of-the-art results on multiple benchmarks. Later works such as DaSiam (Zhu et al., 2018), C-RPN (Fan and Ling, 2019) and SiamRPN++ (Li et al., 2019) improve upon SiamRPN. However, since anchors are introduced for region proposal generation, these trackers can be sensitive to the numbers, sizes and aspect ratios of anchor boxes, and expertise on hyper-parameter tuning is crucial to obtain successful tracking results with these trackers.

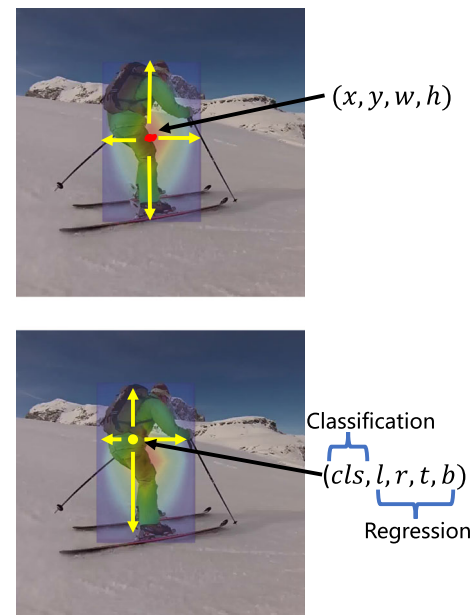


Fig. 1 Location of a bounding box at the central pixel (*upper*) vs. location of a bounding box at each target pixel (*lower*). As shown in the *upper* image, a central location (x, y) with the respecting width and height (w, h) denotes the target bounding box. Limited supervised information is supplied by this definition. The *lower* image shows that by denoting a 4D distance vector (l, r, t, b) , each target location can devote to the bounding box. Notice that $l + r$ and $t + b$ represent the predicted width and height of the target. By this definition, per-pixel supervised information including its foreground confidence cls and its distances (l, r, t, b) to the four sides of the target bounding box can be used in tracking

In this paper, we investigate eliminating the above-mentioned issues of the Siamese architecture based trackers in an effective anchor-free fashion. The question that we want to answer here is that, can we solve the tracking problem by directly regressing the target bounding box in an end-to-end manner? Existing methods like GOTURN (Held et al., 2016) have verified the feasibility of the end-to-end regression approach, with limited success in terms of tracking accuracy. Also, when implementing with deeper backbone networks, training is typically difficult to converge. Our observation is that the information supplied by a bounding box is not sufficiently rich. As shown in Fig. 1, if just locating the bounding box at the center of the object, only one bounding box with the central pixel would be considered in the search region. This will result in limited supervision. To address this issue, all pixels in the search region should be taken into account to contribute to the prediction. For each pixel, we can investigate its target category, and its distances to the four sides of the target bounding box. Motivated by this, we reformulate the visual tracking task into two subproblems: the classification task for pixel category prediction and regression for object bounding box estimation at this pixel. With such decomposition, the tracking task can be solved

in a per-pixel prediction manner. We then design a simple, yet effective Siamese architecture based fully convolutional classification and regression network, termed SiamCAR, to learn the classification and regression models simultaneously in an end-to-end manner.

Essentially, the tracking model consists of two branches. One classification branch aims to predict a category label for each spatial location, and one regression branch focuses on prediction of the location of a bounding box using regression. Different from RPN models (Li et al., 2018; Zhu et al., 2018; Li et al., 2019), which use two response maps for region proposal detection and regression respectively, the proposed model uses one unique response map to predict object location and bounding box directly. Existing cross-correlation (Li et al., 2019, 2018; Bertinetto et al., 2016) operations take the template feature as a whole for global similarity matching to obtain the response map, which leads to the desirable fact that the response of the target center is least affected by the background. As a result, the center pixel of a target tends to offer the most accurate regression result, while pixels far away from the center of a target tend to produce low-quality predicted bounding boxes. To deal with this issue, a localization quality estimation (LQE) branch is associated with the classification branch to provide accurate ranking scores that can benefit the target center prediction and improve tracking performance. Based on this, a coarse-to-fine bounding box tracking strategy is designed to boost the final prediction.

SiamCAR adopts the strategy of online training and offline tracking. The framework is a fully convolutional network with pixel-wise prediction, which is very simple in design, yet powerful in performance. SiamCAR is both anchor and proposal free, thus the number of hyper-parameters being significantly reduced. Without bells and whistles, the proposed tracker achieves the state-of-the-art tracking performance in terms of both accuracy and speed. The work here is an extension of our previous work (Guo et al., 2020). In comparison with the previous version, the new contributions are as follows.

1. We now propose a new coarse-to-fine tracking strategy, which replaces the original average top-k tracking in the previous version (Guo et al., 2020), to further improve the tracking accuracy.
2. We have conducted experiments on more benchmark datasets including TrackingNet (Muller et al., 2018), OTB-100 (Wu et al., 2015) and VOT2020 (Matej et al., 2021). We have also included more comparisons with new state-of-the-art trackers.
3. We have provided more thorough discussions and details in this version.

2 Related Work

In this section, we mainly review the most relevant work. Specifically, Siamese architecture based trackers are discussed since they dominate the tracking performance in recent years. Some related fully convolutional networks with pixelwise prediction are also reviewed.

2.1 Tracking by Siamese Architectures

As one of the pioneering works, GOTURN (Held et al., 2016) trains a Siamese convolutional network to regress the bounding box directly. Since GOTURN tracks via an offline learned generic relationship between the object's appearance and motion, it is difficult to handle the unseen "motions" that are not covered by training samples. Different from GOTURN, SiamFC (Bertinetto et al., 2016) proposes to solve the tracking problem by learning similarity between the template and search region. Inspired by its success, researchers have followed the work and proposed updated models (Dong and Shen, 2018; Valmadre et al., 2017; He et al., 2018; Guo et al., 2017). CFNet (Valmadre et al., 2017) 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 (Guo et al., 2017) proposes to learn a dynamic Siamese network, which achieves better tracking accuracy with acceptable tradeoff of speed. To enhance the model discrimination and adaptiveness, RASNet (Wang et al., 2018) introduces the attention mechanism to adapt the tracking model without online updating. SA-Siam (He et al., 2018) 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 which can be time-consuming.

Inspired by the region proposal network for object detection (Ren et al., 2015), the SiamRPN (Li et al., 2018) tracker performs the region proposal extraction using the output of a 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 (Zhu et al., 2018) 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. Although these aforementioned approaches modified the original SiamFC (Bertinetto et al., 2016) on many aspects, the performance stall mainly because the backbone network used

(AlexNet) is weak. SiamRPN++ (Li et al., 2019) replaces AlexNet with ResNet-50 (He et al., 2016). Meanwhile, it randomly shifts the object location in the search region during training to eliminate the center bias. Such simple tricks surprisingly boost tracking accuracy.

Anchors are adopted in these RPN based trackers for generating region proposals. Besides, anchor boxes can make use of the deep feature maps and avoid repeated computation, which can significantly speed up the tracking process. State-of-the-art trackers, such as SPM (Wang et al., 2019) and SiamRPN (Li et al., 2018), can work in a very high frame rate. Although SiamRPN++ (Li et al., 2019) adopts a very deep neural network, it still works at a real-time speed. Nevertheless, the performance of anchor-based trackers can sometimes be very sensitive to the hyper-parameters of anchors, which need to be carefully tuned to achieve ideal performance. Moreover, since the size and the aspect ratio of anchor boxes are fixed, even with tuned parameters, these trackers still face difficulty in processing objects with large shape deformation and pose variation. Moreover, the design of anchors is also tightly related to the object category. Thus, fixing the design of anchors will make the tracker less generic in terms of tracking multiple categories of objects.

2.2 Fully Convolutional Networks for Pixelwise Prediction

In recent years, researchers demonstrate anchor-free detection frameworks achieving improved performance in object detection and multi-object tracking (Duan et al., 2019; Law and Deng, 2018; Zhou et al., 2019, 2020). These anchor-free detectors like CenterNet (Duan et al., 2019) and CornetNet (Law and Deng, 2018) both can be seen as corner keypoints based detection methods. However, they are not effectively explored for Siamese based single object tracking since spatial information about the corners cannot be explicitly encoded in the response map. Recently, fully convolutional networks (FCNs) that solve vision problems in a pixelwise prediction manner have extensively studied in semantic segmentation (Tian et al., 2019a; Shelhamer et al., 2017; Long et al., 2015) and object detection (Tian et al., 2019b; Huan et al., 2015; Dai et al., 2016). Long et al. (2015) first show that an end-to-end FCN achieves great success in semantic segmentation. Then, researchers leverage the framework for object detection tasks. DenseBox (Huan et al., 2015) shows an end-to-end FCN framework which directly predicts the object class confidences and bounding boxes through all locations of an image. Tian et al. (2019b) propose a one-stage FCN framework termed FCOS for generic objection detection, which eliminates both anchor boxes and proposals, yet achieving comparable detection accuracy. The key insight of such an FCN framework for object detection is predicting a class category and a localization vector for each spatial loca-

tion on a feature map, where the localization vector represents the relative distances to the four sides of the target bounding box at this location. Accordingly, our motivation for solving the visual tracking problem meets the core of FCN.

In this paper, we show that the above-mentioned issues of Siamese based trackers can be greatly alleviated with a carefully designed Siamese fully convolutional framework in an end-to-end per-pixel prediction manner. Moreover, we demonstrate that a tracker with a much simpler structure can achieve even better performance than previous state-of-the-art methods.

3 Our Method

We now introduce our SiamCAR network in detail. As mentioned, we decompose the tracking task into two sub-problems as classification and regression, and then solve them in a per-pixel prediction manner. The proposed framework mainly consists of two simple subnetworks. The first one is a Siamese network adopted for feature extraction, and the second one is the classification and regression subnetwork for direct bounding box prediction. As show in Fig. 2, the entire system can be implemented as a neat fully convolutional network, simple and easy to interpret. In Section 3.1, we introduce the designs of the subnetwork for feature extraction. In Section 3.2, we develop algorithms for target bounding box prediction with classification for pixel category and regression for bounding box at this pixel. Finally, we introduce the coarse-to-fine tracking strategy to boost the final prediction.

3.1 Feature Extraction

Here, we take advantage of the Siamese architecture with convolutional neural networks to construct a Siamese subnetwork for 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 outputs 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 (Bertinetto et al., 2016). Since we need to decode the response map R in the subsequent bounding box prediction subnetwork to obtain the location and scale information of the target, we hope that the response map R can retain abundant information association. However, the cross-correlation layer can only generate a single-channel compressed response map, which lacks useful features and important information for latter decoding, as suggested by Li et al. (2019) that different feature channels typically take

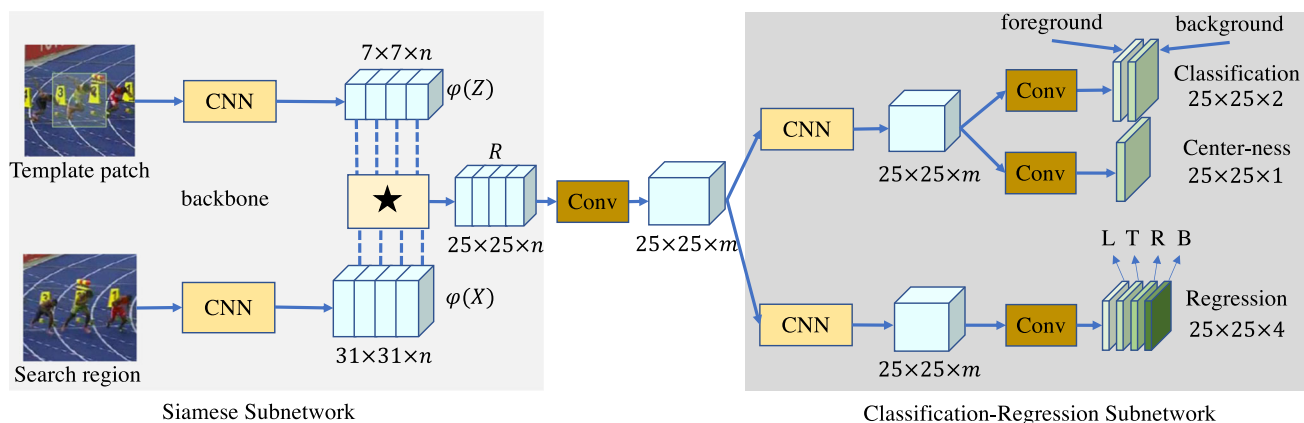


Fig. 2 Illustration of the proposed framework SiamCAR: The left side is a Siamese subnetwork with a depth-wise cross correlation layer (denoted by \star) for multi-channel semantic response map extraction. The right side shows the classification and regression subnetwork for bounding box prediction, which is taken to decode the location and scale information of the object from multi-channel response map. Given a video sequence, the object denoted in the first frame is served as the template patch, which is fixed and unupdated during the whole tracking period. For a search region in the current frame, its target bounding box can be depicted by a location with relative 6D vector $B = (cls, cen, l, t, r, b)$, while cls represents the foreground confi-

dence score output by the foreground-background classification branch, and (l, r, t, b) represent the distances to the four sides of the bounding box output by the distance regression branch. Here cen represents the center-ness score output by the attached center-ness branch, which is adopted to measure the distance between this location and the target center. Meanwhile, a coarse-to-fine tracking strategy is introduced to calculate the final tracking result through the output predictions of the proposed framework. Note that the framework can be implemented as a fully convolutional network, which is simple, neat and easy to interpret

distinct semantic information. Inspired by them, we also use a depth-wise correlation layer to produce multiple semantic similarity maps:

$$R = \varphi(X) \star \varphi(Z), \quad (1)$$

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

Since CNN features are extracted with multiple layers, these layers encode features of different aspects with different levels. Low-level features such as edge, corner, color and shape that represent better visual attributes are indispensable for localization, while high-level features are better to represent semantic attributes that are crucial for discrimination. Thus, only using the feature of the last layer may lead to deviation. Many methods take advantage of fusing both low-level and high-level features to improve the tracking accuracy (Ma et al., 2018a; Li et al., 2019; Wang et al., 2019). Here we also consider to aggregate multi-layer deep features for tracking. We use the modified ResNet-50 to be the backbone network, same as in SiamRPN++ (Li et al., 2019). 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:

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

where $\mathcal{F}_{i=3,4,5}(X)$ 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 obtain a multi-channel response map. The response map is then convolved 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 for bounding box prediction.

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 convolutional 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 foreground-background classification branch to predict the

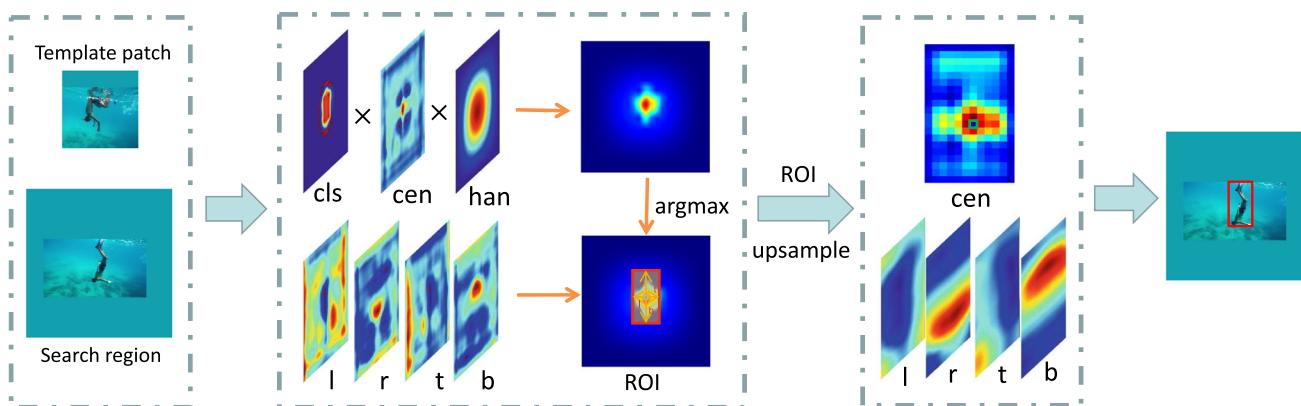


Fig. 3 Illustration of the coarse-to-fine tracking strategy. For the current tracking frame, the proposed classification-regression subnetwork outputs six maps: the *cls* map, the *cen* map, and the *l*, *r*, *t*, *b* maps. During the coarse detection operation, a Hanning map *han* output by a cosine window is introduced to boost the center bias. By mapping the coarse

detection into the response map, a ROI can be obtained. During the refinement operation, the center location is computed by up-sampling the corresponding ROI region of the *cen* and *l*, *r*, *t*, *b* maps to the same resolution of the search region. The final target bounding box is achieved by this center location and its relative distance vector (*l*, *r*, *t*, *b*)

category for each location, and a distance regression branch to estimate the target bounding box at this location (see Fig. 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 $A_{w \times h \times 2}^{cls}$ and the regression branch outputs a regression feature map $A_{w \times h \times 4}^{reg}$. Here *w* and *h* represent the width and the height of the extracted feature maps respectively. As that shown in Fig. 2, each point $(i, j, :)$ in $A_{w \times h \times 2}^{cls}$ 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 $A_{w \times h \times 4}^{reg}$ contains a 4D distance 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 $A_{w \times h \times 4}^{reg}(i, j, :)$ can be calculated by:

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

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

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

where L_{IoU} is the IoU loss as in Yu et al. (2016) and $\mathbb{I}(\cdot)$ is an indicator function defined by:

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

The target center plays an important role for locating a target bounding box. An observation is that the locations far away from the center of a target tend to produce low-quality predicted bounding boxes, which reduces the performance of the tracking system. This is because the more far away from the center, the more significant the background interference is. Therefore, we investigate to add a localization quality estimation (LQE) branch associated with the classification branch. The output of this branch should depict the attribute of a location in relate to the target center. There are many algorithms to achieve this effect, such as the Gaussian distribution with the target center as the mean (Zhou et al., 2019, 2020), or the center-ness algorithm presented in FCOS (Tian et al., 2019b). Here we take the center-ness algorithm to produce the localization quality estimation. As shown in Fig. 2, the branch outputs a center-ness feature map $A_{w \times h \times 1}^{cen}$, where each point value gives the center-ness score of the corresponding location. The score $C(i, j)$ in $A_{w \times h \times 1}^{cen}(i, j)$ 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 target center in the search region. If (x, y) is a location within background, the value of $C(i, j)$ is set to 0. The center-ness loss can be computed as

$$\mathcal{L}_{cen} = \frac{-1}{\sum \mathbb{I}(\tilde{t}(i, j))} \sum_{\mathbb{I}(\tilde{t}(i, j))=1} C(i, j) * \log A_{w \times h \times 1}^{cen}(i, j) + (1 - C(i, j)) * \log(1 - A_{w \times h \times 1}^{cen}(i, j)). \quad (7)$$

A visual description of the center-ness score for each pixel in the bounding box is given in Fig. 1. By introducing the center-ness branch, the center of an object can be better predicted, and the outliers can be eliminated.

To sum up, the overall loss function is

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

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

3.3 The Coarse-to-Fine Tracking Phase

Tracking aims at predicting a bounding box for the target in the current frame. To decode the object status from the response map, for each location (i, j) , the proposed classification-regression subnetwork produces a 6D vector $T_{ij} = (cls, cen, l, t, r, b)$, where cls represents the foreground score of classification, cen represents the center-ness score, and $l + r$ and $t + b$ represent the predicted width and height of the target in the current frame. In the real tracking process, we observed that the jitter error may appear between adjacent frames if only use Non-Maximum Suppression to estimate the target bounding box. The reason is that the most accurate regression result should be given by the location of the target center, of which the response is least affected by the background. Since our model solves the object tracking with a per-pixel prediction manner, each location is relative to a predicted bounding box. Any inaccurate value may lead to prediction errors of the target center and scale. To eliminate the problem, we adopt a coarse-to-fine strategy to estimate the target center, achieving much stable and accurate bounding box.

Coarse detection During tracking, the size and aspect ratio of the bounding box typically see minor change across consecutive frames. To supervise the prediction using this spatio-temporal consistency, we adopt a scale change penalty p_{ij} as similar to that in Li et al. (2018; 2019), Fan and Ling (2019) to re-rank the foreground 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:

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

where H is the cosine window and λ_d is a balance weighting scalar ($0 \leq \lambda_d \leq 1$). The penalty and cosine window can discourage large displacement and large scale change in size and ratio during the tracking phase, thus alleviating the influence of distractors. However, they also produce a side effect that the output q with the highest score may be deviated from the target center. Consequently, the queried location q may be a target pixel near the center but not the center point itself. Accordingly, we take its l, t, r, b to determine a coarse bounding box B_{coarse} , within which the target center is estimated by the following region refinement operation.

Refinement As mentioned before, each location (i, j) in the response map R^* can be mapped back onto the input search region as (x_i, y_j) . Let $RoI = \{(i, j) | (x_i, y_j) \in B_{coarse}\}$ denotes a region of interest, obviously the resolution of pixels in RoI is $1/stride$ of those in B_{coarse} . In the proposed architecture, the stride is set as 8. Then, we can apply the Bicubic Interpolation with a scale 8 to up-sample the corresponding RoI region of maps A^{cen} and A^{reg} . Thus, the up-sampled maps A_{up}^{cen} and A_{up}^{reg} will have the same resolution as the search region. The location with the highest value in A_{up}^{cen} is considered as the target center prediction of the center-ness branch, that is:

$$q_{up}^* = \arg \max_{i, j} \{A_{up}^{cen}\}. \quad (10)$$

By taking a simple linear transformation, we can get its corresponding location (x_{cen}, y_{cen}) on the search region. According to q_{up}^* , the corresponding vector in the map A_{up}^{reg} is $(l_{up}, t_{up}, r_{up}, b_{up})$. It is empirically observed that the vector has the following characteristics: $|l_{up} - r_{up}| \approx 0$ and $|t_{up} - b_{up}| \approx 0$. Therefore, we take use of $l_{up} + r_{up}$ as width of the target and $t_{up} + b_{up}$ as height. The reason is that the values l, r, t, b are predicted by four different channels of the regression branch, their regression errors can be regarded as independent to each other. Thus, the regression error can be averaging reduced through this simple operation. By refinement, the final output bounding box of the tracking model is:

$$B_{fine} = (x_{cen}, y_{cen}, l_{up} + r_{up}, t_{up} + b_{up}) \quad (11)$$

where (x_{cen}, y_{cen}) represents the center location of the object, $l_{up} + r_{up}$ represents the weight and $t_{up} + b_{up}$ represents the height of the bounding box.

By leveraging the coarse-to-fine strategy, the proposed framework can output a more accurate and stable final tracking result. Ablation studies demonstrate the effectiveness of this tracking strategy. The whole tracking phase is intuitively shown in Fig. 3.

4 Experiments and Results

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 SiamRPN++ (Li et al., 2019), respectively to 127 pixels and 255 pixels. In fact, the proposed framework can work with both shallow and deeper backbone architectures (shown in the ablation study in Section 5.2). In this section, we adopt the modified ResNet-50 in SiamRPN++ (Li et al., 2019) as the backbone to conduct the comparison experiments. The network is pretrained on ImageNet (Russakovsky et al., 2015). 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 the COCO (Lin et al., 2014), ImageNet DET, ImageNet VID (Russakovsky et al., 2015) and YouTube-BB (Real et al., 2017) datasets for experiments on the UAV123 (Muller et al., 2016), OTB-100 (Wu et al., 2015) and VOT2020 (Matej et al., 2021) datasets. For experiments on the TrackingNet (Muller et al., 2018) dataset, we replace YouTube-BB (Real et al., 2017) with the TrackingNet training set to train the model. It should be noticed that for experiments on the GOT-10K (Huang et al., 2018) and LaSOT (Fan et al., 2019) datasets, 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 the initial 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 Fig. 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 coarse-to-fine strategy introduced in Section 3.3 is adopted to compute the final tracking result, and the queried location q refers to the coarse predicting bounding box. 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 (Huang et al., 2018) 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-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. 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 against state-of-the-art trackers including SiamRPN++ (Li et al., 2019), SiamRPN (Li et al., 2018), ATOM (Danelljan et al., 2019), SPM (Wang et al., 2019) and other baselines or state-of-the-art approaches. Figure 4 shows the success plots of all comparing trackers. The quantitative results of trackers on the average overlap and success rate are listed in Table 1, where the Top-2 results are highlighted in italic and bold respectively. Overall, the proposed tracker performs best in terms of all metrics. Our SiamCAR surpasses the base-

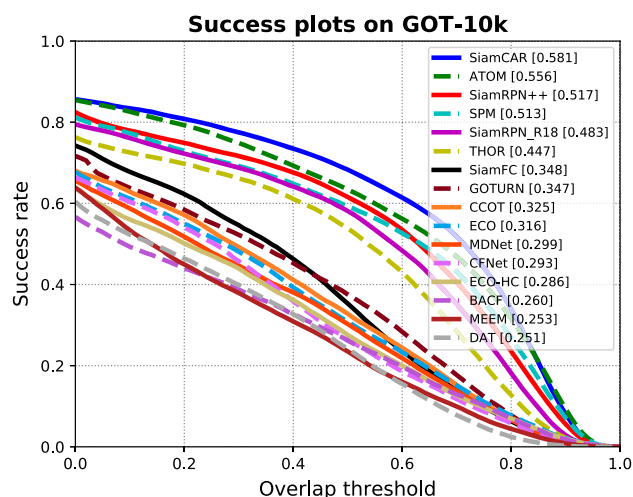


Fig. 4 Success plots on the GOT-10K (Huang et al., 2018) dataset. Our SiamCAR significantly outperforms the baselines and other state-of-the-art methods

Table 1 Evaluation on the GOT-10K (Huang et al., 2018) dataset

Tracker	<i>AO</i>	<i>SR</i> _{0.5}	<i>SR</i> _{0.75}	FPS	Hardware	Implementation
KCF (Henriques et al., 2014)	0.203	0.177	0.065	94.66	CPU	Matlab
fDSST (Danelljan et al., 2016a)	0.206	0.187	0.075	30.43	CPU	Matlab
SRDCF (Danelljan et al., 2015)	0.236	0.227	0.094	5.58	CPU	Matlab
Staple (Luca et al., 2016)	0.246	0.239	0.089	28.87	CPU	Matlab
SAMF (Li and Zhu, 2014)	0.246	0.241	0.084	7.43	CPU	Matlab
DSST (Danelljan et al., 2014)	0.247	0.223	0.081	18.25	CPU	Matlab
DAT (Pu et al., 2018)	0.251	0.242	0.048	45.52	CPU	Matlab
MEEM (Zhang et al., 2014)	0.253	0.235	0.068	20.59	CPU	Matlab
BACF (Kiani et al., 2017)	0.260	0.262	0.101	14.44	CPU	Matlab
CFnet (Valmadre et al., 2017)	0.293	0.265	0.087	35.62	Titan X	Matlab
MDnet (Nam and Han, 2016)	0.299	0.303	0.099	1.52	Titan X	Python
ECO (Danelljan et al., 2017)	0.316	0.309	0.111	2.62	CPU	Matlab
CCOT (Danelljan et al., 2016b)	0.325	0.328	0.107	0.68	CPU	Matlab
SiamFC (Bertinetto et al., 2016)	0.374	0.404	0.144	25.81	Titan X	Matlab
SPM (Wang et al., 2019)	0.513	0.593	0.359	72.30	Titan Xp	Python
SiamRPN++ (Li et al., 2019)	0.517	0.616	0.325	49.83	RTX 2080 Ti	Python
ATOM (Danelljan et al., 2019)	0.556	0.634	0.402	20.71	GTX-1050	Python
SiamCAR	<i>0.581</i>	<i>0.683</i>	<i>0.441</i>	52.20	RTX 2080ti	Python

The average overlap and success rate of baselines and state-of-the-art trackers are evaluated on the GOT-10K (Huang et al., 2018) dataset. Top-2 results are highlighted in italic and bold respectively

line SiamRPN++ (Li et al., 2019) by 6.4% in *AO*, 6.7% in *SR*_{0.5} and 11.6% in *SR*_{0.75}. Compared with the second-best tracker ATOM (Danelljan et al., 2019), SiamCAR improves the scores by 2.5%, 4.9% and 3.9% respectively for *AO*, *SR*_{0.5} and *SR*_{0.75}.

In Table 1, we also show the tracking running time in frame-per-second (FPS). The reported speeds of SiamRPN++ and SiamCAR 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 52.20 FPS. Compared with SiamRPN++, SiamCAR improves the tracking speed by 2.37 FPS.

Figure 5 shows some qualitative comparison results with three popular trackers SiamRPN++ (Li et al., 2019), SPM (Wang et al., 2019), ECO (Danelljan et al., 2017) and the proposed algorithm on some challenging sequences from the GOT-10K dataset. The anchor-free tracker ECO can work in dull background (*GOT-10K_Test_140*) but the performance is still much worse than the anchor-based trackers SiamRPN++ (Li et al., 2019) and SPM (Wang et al., 2019). It quickly drifts to the background when the background is cluttered and similar distractors appears (*GOT-10K_Test_91*, *GOT-10K_Test_92* and *GOT-10K_Test_81*). The anchor-based methods SiamRPN++ (Li et al., 2019) and SPM (Wang et al., 2019) fail to track the whole object when the object suffer from large shape deformation and

pose variation (*GOT-10K_Test_81*). SPM (Wang et al., 2019) may also easy to drift to the background with similar appearance (*GOT-10K_Test_86*). With per-pixel classification and regression, the proposed bounding box prediction subnetwork can decode sufficient category information and spatial details against the shape and appearance changes caused by background clutters, fast motion and rotations (*GOT-10K_Test_91*, *GOT-10K_Test_60*, *GOT-10K_Test_86*, *GOT-10K_Test_81* and *GOT-10K_Test_140*). Since the size and aspect ratio are regressed by the model instead of prefixed anchors, the proposed framework can greatly alleviate drastic spatial scale changes caused by large shape deformation and rotation, performing well on locating the whole object with a much accurate enclosing bounding box (*GOT-10K_Test_81*, *GOT-10K_Test_86* and *GOT-10K_Test_140*).

4.3 Results on LaSOT

LaSOT (Fan et al., 2019) 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.



Fig. 5 Examples of tracking results. We show some visual tracking results of three popular trackers SiamRPN++ (Li et al., 2019), SPM (Wang et al., 2019), ECO (Danelljan et al., 2017) and the proposed algorithm on six challenging sequences from GOT-10K (from top to bottom: *GOT-10K_Test_60*, *GOT-10K_Test_86*, *GOT-10K_Test_91*,

GOT-10K_Test_92, *GOT-10K_Test_81*, and *GOT-10K_Test_140*). Our SiamCAR can accurately predict the bounding boxes even when objects suffer from rotations, background clutters, large scale variation and large pose variation, while SiamRPN++ and SPM give much rougher results and ECO drifts to the background

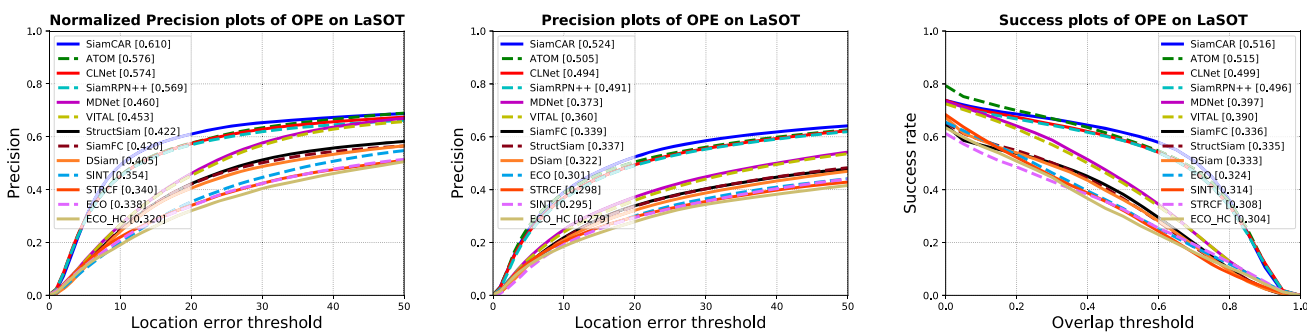


Fig. 6 Normalized precision plots, precision plots of OPE and success plots of OPE on the LaSOT (Fan et al., 2019) dataset. Our SiamCAR significantly outperforms baselines and the state-of-the-art methods

We compare our SiamCAR with state-of-the-art trackers and baselines including SiamRPN++ (Li et al., 2019), ATOM (Danelljan et al., 2019), CLNet (Dong et al., 2020), DSiam (Guo et al., 2017) and other baselines. As shown in Fig. 6, our SiamCAR achieves the best performance. Com-

pared with SiamRPN++, our SiamCAR improves the scores by 4.1%, 3.3% and 2.0% respectively for normalized precision plots, precision plots of OPE and success plots of OPE. Compared with ATOM (Danelljan et al., 2019), our Siam-

Table 2 Comparisons on the TrackingNet (Muller et al., 2018) dataset

Method	Success	Norm Precision	Precision
SiamFC (Bertinetto et al., 2016)	55.9	65.2	51.8
CFNet (Valmadre et al., 2017)	57.8	65.4	53.3
ECO (Danelljan et al., 2017)	55.4	61.8	49.2
MDNet (Nam and Han, 2016)	60.6	70.5	56.5
DaSiamRPN (Zhu et al., 2018)	63.8	73.3	59.1
ATOM (Danelljan et al., 2019)	70.3	77.1	64.8
GlobalTrack (Lianghua Huang, 2020)	70.4	75.4	65.6
ROAM++ (Yang et al., 2020)	67.0	75.4	62.3
D3S (Lukezic et al., 2020)	72.8	76.8	66.4
SiamRPN++ (Li et al., 2019)	73.3	80.0	69.4
SiamCAR (Ours)	<i>74.01</i>	<i>80.44</i>	68.41

The success rate, precision and normalized precision of baselines and state-of-the-art trackers are evaluated on the TrackingNet (Muller et al., 2018) dataset. Top-2 results are highlighted in italic and bold respectively

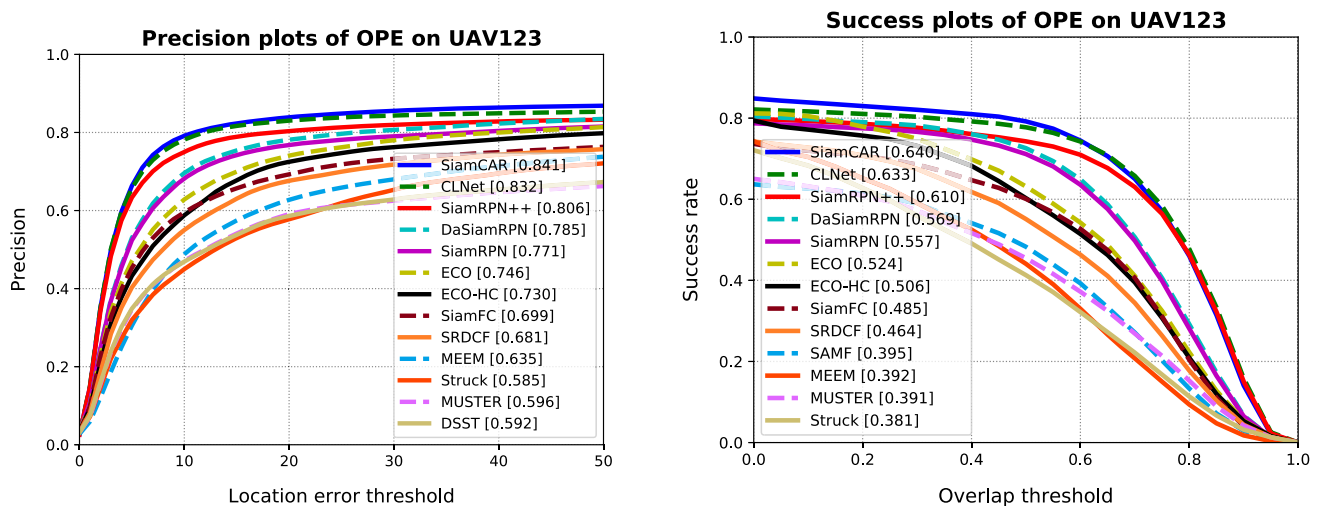


Fig. 7 Precision plots of OPE and success plots of OPE on the UAV123 (Muller et al., 2016) dataset. Our SiamCAR performs the best on both evaluation metrics

CAR improves by over 3.4%, 1.9% and 0.1% respectively for the three metrics.

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

4.4 Results on TrackingNet

TrackingNet (Muller et al., 2018) is a large-scale dataset and benchmark for object tracking in the wild. It provides over 30,000 video sequences for training and 511 video sequences for testing. The evaluation metrics include success rate, precision and normalized precision. We compare our SiamCAR with ten state-of-the-art approaches and baselines including D3S (Lukezic et al., 2020), ROAM++ (Yang et al., 2020), SiamRPN++ (Li et al., 2019), GlobalTrack (Lianghua Huang, 2020), ATOM (Danelljan et al., 2019) and

other baselines. As shown in Table 2, our SiamCAR surpasses state-of-the-art trackers like D3S, GlobalTrack and ATOM in all three metrics. Compared with SiamRPN++ (Success score of 73.3%, Precision score of 69.4% and Normalized Precision score of 80%), our SiamCAR achieves a comparable result (Success score of 74.01%, Precision score of 68.41% and Normalized Precision score of 80.44%) with a much simpler network.

4.5 Results on OTB100

OTB-100 (Wu et al., 2015) contains 100 sequences with substantial variations that collected and annotated from the commonly used tracking sequence. The test sequences are manually tagged with 11 attributes to represent the challenging aspects, including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane

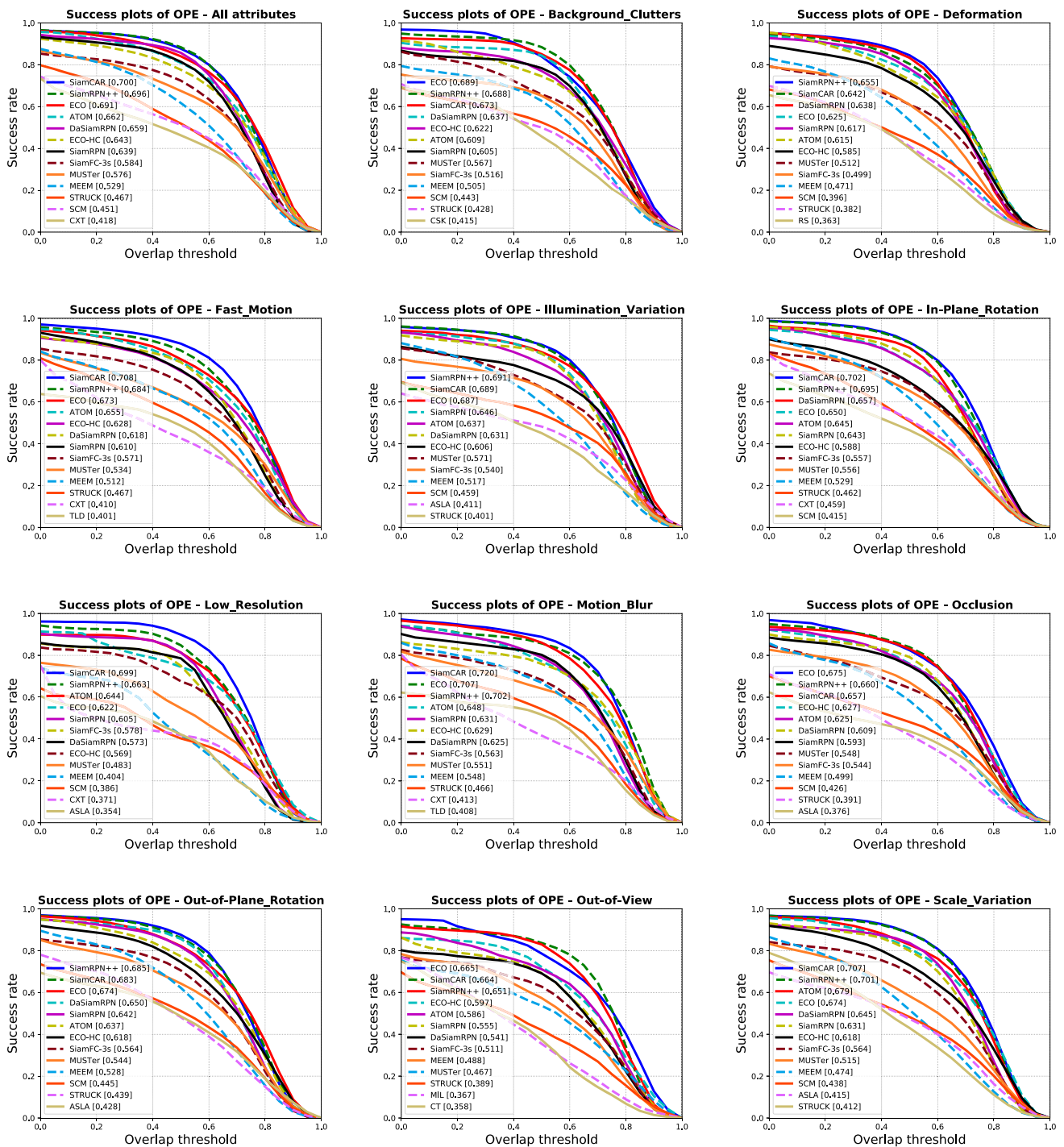


Fig. 8 Success plots of OPE on the OTB-100 (Wu et al., 2015) dataset with challenging attributes including background clutters, deformation, fast motion, illumination variation, in-plane rotation, low resolution,

rotation, out-of-plane rotation, out-of-view, background clutters and low resolution. We compare our network with state-of-the-art approaches and baselines including SiamRPN++ (Li et al., 2019), SiamRPN (Li et al., 2018), ATOM (Danell-

jan et al., 2019) and ECO (Danelljan et al., 2017). In Fig. 8 we show an evaluation on success plots of *OPE* for 11 annotated attributes and the overall attributes. It should be noticed that the results here reported by SiamRPN++ (Li et al., 2019)

Table 3 Comparisons on the VOT2020 (Matej et al., 2021) dataset

	VOT-ST2020			VOT-RT2020		
	EAO	A	R	EAO	A	R
IVT (Ross et al., 2008)	0.092	0.345	0.244	0.089	0.349	0.229
KCF (Henriques et al., 2014)	0.154	0.407	0.432	0.154	0.406	0.434
SiamFC (Bertinetto et al., 2016)	0.179	0.418	0.502	0.172	0.422	0.479
CSR-DCF (Lukežič et al., 2018)	0.193	0.406	0.582	0.193	0.405	0.565
SiamRPN++ (Li et al., 2019)	0.244	0.443	0.672	0.244	0.443	0.673
ATOM (Danelljan et al., 2019)	0.271	0.462	0.734	0.237	0.440	0.687
SiamCAR	0.273	0.449	0.732	0.272	0.449	0.732

The Expected Average Overlap (EAO), Accuracy (A) and Robustness (R) of baselines and state-of-the-art trackers are evaluated on the VOT2020 (Matej et al., 2021) dataset. Top-2 results are highlighted in italic and bold, respectively

are tracked by a specific trained model for the OTB dataset, while our results are reported by a general trained model. As shown in Fig. 8, the proposed SiamCAR ranks first in terms of the overall attributes. Especially, our SiamCAR significantly improves the tracking accuracy against the impacts of low resolution, fast motion, motion blur and scale variation, which benefits from the accurate decoded target information by our classification-regression subnetwork. Compared with state-of-the-art RPN trackers (Li et al., 2019; Zhu et al., 2018; Li et al., 2018), our SiamCAR obtains competitive results with a much simpler network, and it does not require tuning parameters heuristically.

4.6 Results on UAV123

UAV123 (Muller et al., 2016) 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 state-of-the-art approaches and baselines including CLNet (Dong et al., 2020), SiamRPN++ (Li et al., 2019), SiamRPN (Li et al., 2018), SiamFC (Bertinetto et al., 2016) and ECO (Danelljan et al., 2017) on this dataset. The success plot and precision plot of OPE are used to evaluate the overall performance here. As shown in Fig. 7, our SiamCAR outperforms all other trackers on both metrics. Compared with SiamRPN++ (Li et al., 2019), our SiamCAR improves the scores by 3.5% and 3.0% respectively for precision plots of OPE and success plots of OPE.

4.7 Results on VOT2020

VOT2020 (Matej et al., 2021) dataset contains 60 public sequences with different challenging factors. The evaluation on VOT2020 has two settings: the VOT short-term tracking challenge VOT-ST2020 under appearance variation, occlu-

sion and clutter, as well as the VOT short-term real-time challenge VOT-RT2020 under time constraints. Following the evaluation protocol of VOT2020, the Expected Average Overlap (EAO) is taken as the main ranking metric, and evaluation in terms of Accuracy (A) and Robustness (R) are also reported.

We compare our SiamCAR with state-of-the-art approaches and baselines including ATOM (Danelljan et al., 2019), SiamRPN++ (Li et al., 2019), SiamFC (Bertinetto et al., 2016) and CSR-DCF (Lukežič et al., 2018) on this dataset. As shown in Table 3, our SiamCAR achieves leading performance on the dataset. Compared with SiamRPN++ (Li et al., 2019), our SiamCAR improves the scores by at least 2.8%, 0.6% and 5.9% respectively for EAO, Accuracy and Robustness. Compared with the online model updating tracker ATOM (Danelljan et al., 2019), our tracker with simpler structure and off-line tracking strategy improves the score of the main ranking metric EAO by 0.2% on the VOT-ST2020 challenge and 3.5% on the VOT-RT2020 challenge. The comparisons indicate that the proposed tracker can achieve much stable performance with real-time request.

4.8 Failure Cases

In Fig. 9, we show the tracking failure instances in extreme cases. For the *upper-left* and *lower-right* frames, when similar distractors with the same category and appearance occur along with fast motions, background clutters and complete occlusions, the proposed method can not successfully discriminate the target person or face due to the high similarity with other people. For the *upper-right* frame, when long-term occlusions occur, our framework cannot perform well since the lack of information for target center location. For the *lower-left* frame, the ground truth is the upper part of the body, while our framework tracks the whole body. The reason is that the appearance of the lower body is similar to the upper body, and our framework tends to detect the person as a whole.

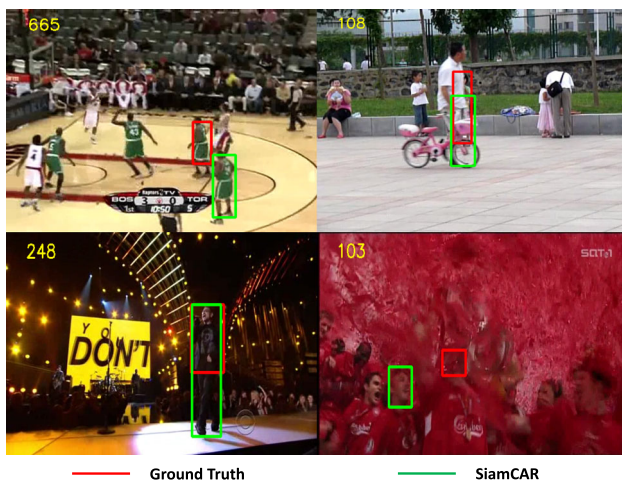


Fig. 9 Failure examples in extreme cases with high similar distractors, severe occlusions and motion blur

Since the method is a simple one-shot learning framework, the performance can be boosted via new modules like attention mechanism or model updating in the future.

5 Ablation Study

In order to validate the impact of different tracking strategy and backbones, some ablation experiments are conducted on popular benchmarks.

5.1 Coarse-to-Fine Tracking Strategy

In this paper, we formulate a coarse-to-fine tracking strategy instead of the top-k average strategy in our previous work (Guo et al., 2020) to compute the final target bounding box. To analysis the impact, we conduct the comparison between the two tracking strategies and the coarse detection on the OTB-100 (Wu et al., 2015), UAV123 (Muller et al., 2016), LaSOT (Fan et al., 2019) and GOT-10K (Huang et al., 2018) datasets.

As shown in Table 4, compared with the coarse detection, both the top-k average strategy and the coarse-to-fine strategy achieve performance improvement on all benchmarks. This is because the queried location q computed by Eq. 9 for coarse detection may be visibly deviated from the target center due to the influence of the scale change penalty p_{ij} and cosine window. Because of the particularity of SOT task that the target is given only in the first frame, small deviation errors are easy to accumulate, leading to the failure tracking of subsequent frames.

Compared with the top-k average strategy, SiamCAR with the new tracking strategy improves the scores in terms of all the indicators. The success on OTB-100 is improved by

4.0% from 0.660 to 0.700, and the precision is increased by 2.4% from 0.890 to 0.914. Indicators on other datasets are also improved by at least 0.9% to 2.6%, which demonstrate that the coarse-to-fine strategy further boost the tracking performance. The reason is that directly using the largest score output by the center-ness branch to determine the target center may yield certain deviation error, since there is a resolution ratio difference between the response map and the search region. By upsampling the center-ness map and the regression map, we can achieve the target center and its bounding box through a Bicubic Interpolation method, which can reduce the deviation caused by resolution change.

5.2 SiamCAR Framework versus the Backbones

In order to verify the effectiveness of the proposed framework with different backbone architectures, we conduct the tracking experiments on the OTB100 and UAV123 datasets by using Alexnet, mobileNet, ResNet-18 and ResNet-50 as backbones. These backbone architectures are different in depth of layers and lead to great variations in computational efficiency. Table 5 shows the tracking performance with the success, precision and frame-per-second (FPS) as metrics. To demonstrate the contribution of our classification-regression subnetwork, comparisons with the most representative baselines are also shown for the shallow network AlexNet, the deeper network ResNet-18 and the deepest network ResNet-50.

With AlexNet, the proposed framework outperforms SiamRPN (Li et al., 2018) by 2.0% in success and 1.6% in precision, as well as 3.4% in success and 1.1% in precision relatively on the OTB100 and UAV123 datasets. With ResNet-50, the proposed framework outperforms SiamRPN++ (Li et al., 2019) by 0.8% in success and 0.5% in precision relatively on the two datasets. A speed of 170 FPS can be achieved with Alexnet. By replacing AlexNet with ResNet-50, the precision increase by around 5.3% and 5.9% relatively on the two datasets while the tracking speed decreases to 52 FPS, which is still in real-time speed. Obviously, the proposed framework can benefit from deeper networks. Notably, SiamCAR with ResNet-18 can achieve a tracking speed of 1.96 times faster than that of SiamRPN++ (Li et al., 2019) with ResNet-50 while producing a better performance on the UAV123 dataset. Compared with the online tracker ATOM (Danelljan et al., 2019), SiamCAR with ResNet-18 surpasses it by 1.0% in success and 1.2% in precision on the OTB100 dataset. Though SiamCAR with ResNet-18 obtains lower performance than ATOM on the UAV123 dataset, it achieves a tracking speed of 3.4 times faster. The reason is that much longer video sequences can obviously benefit online model updating tracker, but the simpler architecture of our proposed tracker can greatly improve the tracking efficiency. Accord-

Table 4 Comparisons with different tracking strategies

Datasets	Indicators	SiamCAR-ctf	SiamCAR-ave (Guo et al., 2020)	SiamCAR-coarse
OTB-100 (Wu et al., 2015)	Success	0.700	0.660	0.652
	Precision	0.914	0.890	0.854
UAV123 (Muller et al., 2016)	Success	0.640	0.620	0.614
	Precision	0.839	0.817	0.806
LaSOT (Fan et al., 2019)	Success	0.516	0.507	0.494
	Norm precision	0.610	0.600	0.589
	Precision	0.524	0.510	0.508
GOT-10K (Huang et al., 2018)	AO	0.581	0.569	0.558
	SR0.5	0.683	0.670	0.657
	SR0.75	0.441	0.415	0.423

Experiments are conducted with different official measurements relative to popular benchmarks. SiamCAR-coarse: the proposed framework with only coarse detection operation; SiamCAR-ctf: the proposed framework with coarse-to-fine strategy; SiamCAR-ave (Guo et al., 2020): the proposed framework with top-k average strategy

Table 5 Comparisons on the OTB100 (Wu et al., 2015) and UAV123 (Muller et al., 2016) dataset with different backbones

Backbone	Method	OTB100		UAV123		FPS
		Success	Precision	Success	Precision	
AlexNet	SiamFC	0.582	0.771	0.485	0.648	86
	SiamRPN	0.637	0.851	0.557	0.771	160
	SiamCAR	0.657	0.867	0.591	0.782	170
MobileNet	SiamCAR	0.676	0.882	0.608	0.802	77
ResNet-18	ATOM	0.663	0.872	0.643	0.856	30
	SiamCAR	0.673	0.884	0.621	0.821	102
ResNet-50	SiamRPN++	0.696	0.915	0.610	0.806	50
	SiamCAR	0.704	0.920	0.640	0.841	52

The success, precision and frame-per-second on the OTB100 (Wu et al., 2015) and UAV123 (Muller et al., 2016) dataset with different backbones are shown in the table. The most representative baselines are listed for respective backbones to validate the contribution of the proposed framework vs. the backbones

ingly, different backbones can be selected to fit the proposed framework to different real tasks, with a trade-off between the tracking accuracy and efficiency.

6 Conclusions

In this paper, we have reformulated the tracking task as two subproblems of classification and regression, thus presented a Siamese classification and regression framework, namely SiamCAR. The proposed framework enables the end-to-end

training of a deep Siamese network for visual tracking. We also show that tracking tasks can be resolved in a per-pixel prediction manner using the proposed neat fully convolution framework. The 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 easily performed next to achieve further improvement.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China (Grant No. 62102364, 62002325, 61802348, 61772268) and the Natural Science Foundation of Jiangsu Province (Grant No. BK20190065). CS and his employer received no financial support for the research, authorship, and/or publication of this article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bertinetto, L., Valmadre, J., Henriques, J., Vedaldi, A., & Torr, P. (2016). Fully-convolutional siamese networks for object tracking. In *Proceedings of European conference on computer vision*.
- Bhat, G., Danelljan, M., Gool, L., & Timofte, R. (2019a). Learning discriminative model prediction for tracking. In *Proceedings of IEEE international conference on computer vision* (pp. 6182–6191).
- Bhat, G., Danelljan, M., Gool, L. V., & Timofte, R. (2019b). Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*.
- Bolme, D., Beveridge, J., Draper, B., & Lui, Y. (2010). Visual object tracking using adaptive correlation filters. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Dai, L., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of advances in neural information processing systems* (pp. 379–387).
- Danelljan, M., Bhat, G., Khan, F., & Felsberg, M. (2017). Eco: Efficient convolution operators for tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Danelljan, M., Bhat, G., Khan, F., & Felsberg, M. (2019). Accurate tracking by overlap maximization. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Danelljan, M., Hager, G., & Khan, F. (2014). Accurate scale estimation for robust visual tracking. In *Proceedings of British machine vision conference*.
- Danelljan, M., Hager, G., Fahad K., & Felsberg, M. (2016a). Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Danelljan, M., Hager, G., Fahad, K., & Felsberg, M. (2015). Learning spatially regularized correlation filters for visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Danelljan, M., Robinson, A., Khan, F., Felsberg, M. (2016b). Beyond correlation filters: learning continuous convolution operators for visual tracking. In *Proceedings of European conference on computer vision*.
- Dong, X., & Shen, J. (2018). Triplet loss in siamese network for object tracking. In *Proceedings of European conference on computer vision*.
- Dong, X., Shen, J., Shao, L., & Porikli, F. (2020). Clnet: A compact latent network for fast adjusting siamese trackers. In *Proceedings of the European conference on computer vision* (pp. 378–395).
- Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6568–6577).
- Fan, H., & Ling, H. (2019). Siamese cascaded region proposal networks for real-time visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., & Ling, H. (2019). Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Gao, J., Zhang, T., & Xu, C. (2019). Graph convolutional tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., & Wang, S. (2017). Learning dynamic siamese network for visual object tracking. In *Proceedings of IEEE international conference on computer vision*.
- Guo, D., Wang, J., Cui, Y., Wang, ZH., & Chen, S. (2020). Siamcar: Siamese fully convolutional classification and regression for visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- He, A., Luo, C., Tian, X., & Zeng, W. (2018). A twofold siamese network for real-time object tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Held, D., Thrun, S., & Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. In *Proceedings of European conference on computer vision* (pp. 749–765). Springer.
- Henriques, J., Caseiro, R., Pedro, M., & Jorge, B. (2014). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Horst, P., Thomas, M., & Horst, B. (2015). In defense of color-based model-free tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Huan, L., Yang, Y., Deng, Y., & Yu, Y. (2015). Densebox: Unifying landmark localization with end to end object detection. arXiv preprint [arXiv:1509.04874](https://arxiv.org/abs/1509.04874).
- Huang, D., Luo, L., Chen, Z., Wen, M., & Zhang, C. (2016). Applying detection proposals to visual tracking for scale and aspect ratio adaptability. *International Journal of Computer Vision*, 524–541.
- Huang, L., Zhao, X., & Huang, K. (2018). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Kiani, G., Hamed, Ashton, F., & Simon, L. (2017). Learning background-aware correlation filters for visual tracking. In *Proceedings of European conference on computer vision*.
- Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of European conference on computer vision* (pp. 6568–6577).
- Li, Y., & Zhu, J. (2014). A scale adaptive kernel correlation filter tracker with feature integration. In *Proceedings of European conference on computer vision*.
- Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

- Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with siamese region proposal network. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Li, F., Yao, Y., Li, P., Zhang, D., Zuo, W., & Yang, M. (2017). Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Lianghua Huang, K. H., Zhao, X. (2020). Globaltrack: A simple and strong baseline for long-term tracking.
- Lin, T., Michael, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., & Zitnick, C. (2014). Microsoft coco: Common objects in context. In *Proceedings of European conference on computer vision*.
- Liu, T., Wang, G., Yang, Q., & Wang, L. (2016). Part-based tracking via discriminative correlation filters. *IEEE Transactions on Circuits and Systems for Video Technology*.
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).
- Luca, B., Jack, V., Stuart, G., Ondrej, M., & HS, TP. (2016). Staple: Complementary learners for real-time tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Lukezic, A., Matas, J., & Kristan, M. (2020). D3s—A discriminative single shot segmentation tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
- Lukežič, A., Vojř, T., Čehovin Zajc, L., Matas, J., & Kristan, M. (2018). Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 126, 671–688.
- Ma, C., Huang, J., Yang, X., & Yang, M. (2018a). Robust visual tracking via hierarchical convolutional features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 2709–2723.
- Ma, C., Huang, J. B., Yang, X., & Yang, M. H. (2018b). Adaptive correlation filters with long-term and short-term memory for object tracking. *International Journal of Computer Vision*, 126, 771–796.
- Matej, K., Ales, L., Jiri, M., Michael, F., Roman, P., & Joni-Kristian, K. (2021). The eighth visual object tracking vot2020 challenge results. In *Proceedings of the European conference on computer vision* (pp. 547–601).
- Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., & Ghanem, B. (2018). Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European conference on computer vision*.
- Muller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for UAV tracking. In *Proceedings of European conference on computer vision*.
- Nam, H., & Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Nam, H., Baek, M., & Han, B. (2016). Modeling and propagating cnns in a tree structure for visual tracking. *arXiv Computer Vision and Pattern Recognition*.
- Pu, S., Song, Y., & Ma, C. (2018). Deep attentive tracking via reciprocal learning. In *Proceedings of advances in neural information processing systems*.
- Real, E., Shlens, J., Mazzocchi, S., Pan, X., & Vanhoucke, V. (2017). Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of advances in neural information processing systems*.
- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 1137–1149.
- Ross, D., Lim, J., Lin, R. S., & Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77, 125–141.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 211–252.
- Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 640.
- Sui, Y., Zhang, Z., Wang, G., Tang, Y., & Zhang, L. (2019). Exploiting the anisotropy of correlation filter learning for visual tracking. *International Journal of Computer Vision*, 127, 1084–1105.
- Tian, Z., He, T., Shen, C., & Yan, Y. (2019a). Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 3126–3135).
- Tian, Z., Shen, C., Chen, H., & He, T. (2019b). Fcos: Fully convolutional one-stage object detection. In *Proceedings of IEEE international conference on computer vision*.
- Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., & Torr, P. (2017). End-to-end representation learning for correlation filter based tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Wang, G., Luo, C., Xiong, Z., & Zeng, W. (2019). Spm-tracker: Series-parallel matching for real-time visual object tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Wang, Q., Zhu, T., Xing, J., Gao, J., Hu, W., & Maybank, S. (2018). Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition* (pp. 4854–4863).
- Wu, Y., Lim, J., & Yang, M. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1834.
- Yang, T., Xu, P., Hu, R., Chai, H., & Chan, A. B. (2020). Roam: Recurrently optimizing tracking model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
- Yu, J., Jiang, Y., Wang, Z., Cao, Z., & Huang, T. (2016). Unitbox: An advanced object detection network. In *ACM international conference on multimedia*.
- Zhang, L., Jagannadan, V., Ponnuthurai, N., Narendra, A., & Pierre, M. (2017a). Robust visual tracking using oblique random forests. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Zhang, J., Ma, S., & Stan, S. (2014). Meem: Robust tracking via multiple experts using entropy minimization. In *Proceedings of European conference on computer vision*.
- Zhang, T., Xu, C., & Yang, M. (2017b). Multi-task correlation particle filter for robust object tracking. In *Proceedings of IEEE conference on computer vision and pattern recognition*.
- Zhou, X., Koltun, V., & Krähenbühl, P. (2020). Tracking objects as points. In *Proceedings of European conference on computer vision* (pp. 474–490).
- Zhou, X., Wang, D., & Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.
- Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., & Hu, W. (2018). Distractor-aware siamese networks for visual object tracking. In *Proceedings of European conference on computer vision*.