Learning Feature Embeddings for Discriminant Model based Tracking

Linyu Zheng[1,2], Ming Tang[1,3], Yingying Chen[1,2,4], Jinqiao Wang[1,2,4], and Hanqing Lu[1,2]

[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[3] Shenzhen Infinova Limited
[4] ObjectEye Inc.
{linyu.zheng, tangm, yingying.chen, jqwang, luhq}@nlpr.ia.ac.cn

Abstract. After observing that the features used in most online discriminatively trained trackers are not optimal, in this paper, we propose a novel and effective architecture to learn optimal feature embeddings for online discriminative tracking. Our method, called DCFST, integrates the solver of a discriminant model that is differentiable and has a closed-form solution into convolutional neural networks. Then, the resulting network can be trained in an end-to-end way, obtaining optimal feature embeddings for the discriminant model-based tracker. As an instance, we apply the popular ridge regression model in this work to demonstrate the power of DCFST. Extensive experiments on six public benchmarks, OTB2015, NFS, GOT10k, TrackingNet, VOT2018, and VOT2019, show that our approach is efficient and generalizes well to class-agnostic target objects in online tracking, thus achieves state-of-the-art accuracy, while running beyond the real-time speed. Code will be made available.

1 Introduction

Visual object tracking is one of the fundamental problems in computer vision. Given the initial state of a target object in the first frame, the goal of tracking is to estimate the states of the target in the subsequent frames [45, 50, 49]. Despite the significant progress in recent years, visual tracking remains challenging because the tracker has to learn a robust appearance model from very limited online training samples to resist many extremely challenging interferences, such as large appearance variation and heavy background clutters. In general, the key problem of visual tracking is how to construct a tracker which can not only tolerate the appearance variation of the target, but also exclude the background interference, while keeping the running speed that is as high as possible.

There has been significant progress in deep convolutional neural networks (CNNs) based trackers in recent years. From a technical standpoint, existing state-of-the-art CNNs-based trackers mainly fall into two categories. (1) The one

is to treat tracking as a problem of similarity learning and is only trained offline. Typically, SINT [42], SiamFC [2], and SiamRPN [31] belong to this category. Although these trackers achieve state-of-the-art performance on many challenging benchmarks, the lack of online learning prevents them from integrating background in an online and adaptive way to improve their discriminative power. Therefore, they are severely affected by heavy background clutters, hindering the further improvement of localization accuracy. (2) The other is to apply CNNs features to the trackers which are discriminatively trained online. Most of these trackers, such as HCF [34], ECO [6], LSART [40], and fdKCF* [48], extract features via the CNNs which are trained on ImageNet [10] for object classification task. Obviously, these features are not optimal for the visual tracking task. Therefore, such trackers are not able to make the visual tracking task sufficiently benefit from the powerful ability of CNNs in feature embedding learning. Even though CFNet [43] and CFCF [15] learnt feature embeddings for online discriminatively trained correlation filters-based trackers by integrating the KCF solver [18] into the training of CNNs, the negative boundary effect [25] in KCF severely degrades the quality of the feature embeddings they learn as well as their localization accuracy. Therefore, it is hard for their architectures to achieve high accuracy in online tracking.

To solve the above problem, in this paper, we propose a novel and effective architecture to learn optimal feature embeddings for online discriminative tracking. Our proposed network receives a pair of images, training image and test image, as its input in offline training [5]. First, an efficient sub-network is designed to extract the features of real and dense samples around the target object from each input image. Then, a discriminant model that is differentiable and has a closed-form solution is trained to fit the samples in the training image to their labels. Finally, the trained discriminant model predicts the labels of samples in the test image, and the predicted loss is calculated. In this way, the discriminant model is trained without circulant and synthetic samples like in KCF, avoiding the negative boundary effect naturally. On the other hand, because it is differentiable and has a closed-form solution, its solver can be integrated into CNNs as a layer with forward and backward processes during training. Therefore, the resulting network can be trained in an end-to-end way, obtaining optimal feature embeddings for the discriminant model-based tracker.

As an instance, we apply the popular ridge regression model in this work to demonstrate the power of the proposed architecture because ridge regression model not only is differentiable and has a closed-form solution, but also has been successfully applied by many modern online discriminatively trained trackers [18, 8, 24, 6, 40, 48] due to its simplicity, efficiency, and effectiveness. In particular, we employ Woodbury identity [37] to ensure the efficient training of ridge regression model when high-dimensional features are used because it allows us to address

[5] In this paper, offline training refers to training deep convolutional neural networks, that is the process of learning feature embeddings, whereas discriminative training refers to training discriminant models, such as ridge regression and SVM. In our approach, each iteration of the offline training involves discriminative training.

the dependence of time complexity on the dimension of features. Moreover, we observed that the extreme imbalance of foreground-background samples encountered during network training slows down the convergence speed considerably and severely reduces the generalization ability of the learned feature embeddings if the commonly used mean square error loss is employed. In order to address this problem, we modify the original shrinkage loss [33] which is designed for deep regression learning and apply it to achieve efficient and effective training.

In online tracking, given the position and size of a target object in the current frame, we extract the features of real and dense samples around the target via the above trained network, *i.e.*, learned feature embeddings, and then train a ridge regression model with them. Finally, in the next frame, the target is first located by the trained model, and then its position and size are refined by ATOM [7].

It is worth mentioning that the core parts of our approach are easy-to-implement in a few lines of code using the popular deep learning packages. Extensive experiments on six public benchmarks, OTB2015, NFS, GOT10k, TrackingNet, VOT2018, and VOT2019, show that the proposed tracker DCFST, *i.e.*, learning feature embeddings with Differentiable and Closed-Form Solver for Tracking, is efficient and generalizes well to class-agnostic target objects in online tracking, thus achieves state-of-the-art accuracy on all six datasets, while running beyond the real-time speed. Fig. 2b provides a glance of the comparison between DCFST and other state-of-the-art trackers on OTB2015. We hope that our simple and effective DCFST will serve as a solid baseline and help ease future research in visual tracking.

2 Related Work

In this section, we briefly introduce recent state-of-the-art trackers, with a special focus on the Siamese network based ones and the online discriminatively trained ones. In addition, we also shortly describe the recent advances in meta-learning for few-shot learning since our approach shares similar insights to theirs.

2.1 Siamese Network Based Trackers

Recently, Siamese network based trackers [42, 2, 47] have received much attention for their well-balanced accuracy and speed. These trackers treat visual tracking as a problem of similarity learning. By comparing the target image patch with the candidate patches in a search region, they consider the candidate with the highest similarity score as the target object. A notable characteristic of such trackers is that they do not perform online learning and update, achieving high FPS in online tracking. Typically, SiamFC [2] employed a fully-convolutional Siamese network to extract features and then used a simple cross-correlation layer to evaluate the candidates in a search region in a dense and efficient sliding-window way. GCT [13] introduced the graph convolution into SiamFC. SiamRPN [31] enhanced the accuracy of SiamFC by adding a region proposal sub-network after the Siamese network. CRPN [12] improved the accuracy of SiamRPN using

a cascade structure. SiamDW [31] and SiamRPN++ [30] enabled SiamRPN to benefit from deeper and wider networks. Even though these trackers achieve state-of-the-art performance on many benchmarks, a key limitation they share is their inability to integrate background in an online and adaptive way to improve their discriminative power. Therefore, they are severely affected by heavy background clutters, hindering the further improvement of localization accuracy.

Different from Siamese trackers, our DCFST can integrate background in an online and adaptive way through training discriminant models. Therefore, it is more robust to background clutters than Siamese trackers.

2.2 Online Discriminatively Trained Trackers

In contrast to Siamese network based trackers, another family of trackers [18, 8, 48] train discriminant models online to distinguish the target object from its background. These trackers can effectively utilize the background of the target, achieving impressive discriminative power on multiple challenging benchmarks. The latest such trackers mainly focus on how to take advantage of CNNs features effectively. HCF [34], ECO [6], fdKCF* [48], and LSART [40] extracted features via the CNNs which are trained on ImageNet for object classification task and applied them to online discriminatively trained trackers. Obviously, these features are not optimal for the visual tracking task. Therefore, these trackers are not able to make the visual tracking task sufficiently benefit from the powerful ability of CNNs in feature embedding learning. In order to learn feature embeddings for online discriminatively trained correlation filters-based trackers, CFNet [43] and CFCF [15] integrated the KCF [18] solver into the training of CNNs. However, it is well known that KCF has to resort to circulant and synthetic samples to achieve the fast training of the filters, introducing the negative boundary effect and degrading the localization accuracy of trackers. Even though CFNet relaxed the boundary effect by cropping the trained filters, its experimental results show that this heuristic idea produces very little improvement. CFCF employed C-COT tracker [9], that is less affected by the boundary effect, in online tracking. However, its offline training does not aim to learn feature embeddings for CCOT, but for KCF, and its running speed is far away from real-time due to the low efficiency of CCOT. Therefore, we argue that it is hard for their architectures to achieve high accuracy and efficiency simultaneously in online tracking.

Different from CFNet and CFCF, our DCFST shares similar insights to DiMP's [4]. Both DCFST and DiMP propose an end-to-end trainable tracking architecture, capable of learning feature embeddings for online discriminatively trained trackers without circulant and synthetic samples. In addition, the training of their discriminant models in both offline training and online tracking are identical. Therefore, they do not suffer from the negative boundary effect and can track target objects more accurately than CFNet and CFCF. The main differences between our DCFST and DiMP are as follows. (1) The architecture of DiMP forces it to use the square-shaped fragments to approximate the real foreground samples, ignoring the actual aspect ratio of the target object. In contrast, that of DCFST is more flexible, allowing us to sample identically to

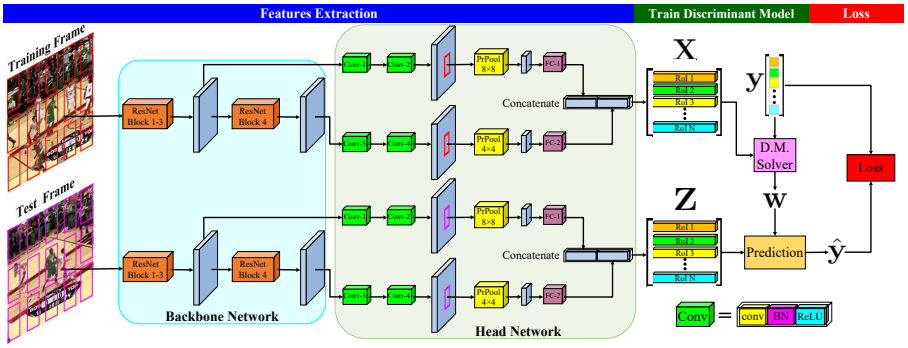


Fig. 1: Full architecture of the proposed network for learning feature embeddings. For each input image, N regions of interest (RoIs) with the target size are generated by uniform sampling. ResNet Block-3 and Block-4 backbone feature maps extracted from the input image are first passed through two convolutional layers to obtain two learned feature maps. Fixed-size feature maps of each RoI are then extracted using PrPool layers and further mapped to feature vectors using fully-connected layers. $\mathbf{X}$ and $\mathbf{Z}$ are data matrices composed of the learned feature vectors of all training and test samples, respectively. A discriminant model $\mathbf{w}$ is trained to fit the samples in $\mathbf{X}$ to their labels. Finally, $\mathbf{w}$ predicts the labels of the samples in $\mathbf{Z}$, and the predicted loss is calculated. Best viewed in color.

the actual size of the target object. Therefore, the foreground samples are approximate in DiMP, but relatively accurate in DCFST. (2) DiMP designed an iterative method to train its discriminant model, which cannot always guarantee an optimal solution. Whereas, DCFST uses a close-form solver which can always guarantee an optimal solution. (3) From the perspective of implementation, DCFST is much simpler than DiMP. The components and codes of the core parts of DCFST are much fewer than those of DiMP. Experiments show that DCFST outperforms CFNet and DiMP in tracking accuracy with large margins and it also outperforms CFCF in both tracking accuracy and speed with large margins.

2.3 Meta-Learning Based Few-Shot Learning

Meta-learning studies what aspects of the learner effect generalization across a distribution of tasks. Recently, differentiable convex optimization based meta-learning approaches greatly promote the development of few-shot learning. Instead of the nearest-neighbor based learners, MetaOptNet [29] used discriminatively trained linear predictors as base learners to learn representations for few-shot learning, and it aimed at learning feature embeddings that generalize well to novel categories under a linear classification rule. Bertinetto *et al.* [1] proposed both closed-form and iterative solvers, based on ridge regression and logistic regression components, to teach a CNN to use standard machine learning tools as part of its internal model, enabling it to adapt to novel data quickly.

To our best knowledge, the proposed DCFST is the first tracker to integrate the solver of a discriminant model that is differentiable and has a closed-form solution into the training of CNNs to learn optimal feature embeddings for online discriminative tracking without circulant and approximate samples. Experiments on multiple challenging benchmarks show that our approach achieves state-of-the-art accuracy at beyond the real-time speed and also sets a simple yet strong baseline for visual tracking. Therefore, we believe that it would promote the development of high-accuracy and real-time tracking.

3 Learning Feature Embeddings

The main task of an online discriminatively trained tracker is to train a discriminant model $\mathbf{w}$ which is able to not only fit the training samples well online, but also generalize well to the test samples. It is well known that not only different modeling methods, such as nearest neighbor and ridge regression, directly effect the generalization ability of $\mathbf{w}$, but features are also crucial to it. Therefore, our approach, DCFST, is developed by designing an architecture to learn optimal feature embeddings for discriminant model-based trackers, rather than more powerful discriminant models as most modern online discriminatively trained trackers did, to improve tracking accuracy.

As shown in Fig. 1, the proposed network receives a pair of 288×288 RGB images, training image and test image, as its input in offline training. It consists of the following three parts: features extraction network, discriminant model solver, and loss function. This section will present them one by one.

3.1 Features Extraction Network

For each input image, the features extraction consists of the following five steps.

(1) N RoIs with the target size are generated by uniform sampling across the whole image. In addition, the vector $\mathbf{y} \in \mathbb{R}^{N\times 1}$ containing their Gaussian labels is constructed as done in KCF [18] with standard deviation of 0.25.
(2) ResNet [16] Block-3 and Block-4 backbone feature maps are extracted from the input image and then passed through two convolutional layers to obtain two learned feature maps. Their strides are 8×8 and 16×16, respectively. Here, all convolutional kernels are 3×3 and all convolutional layers are followed by BatchNorm [20] and ReLU.
(3) Fixed-size feature maps of each RoI are respectively extracted from the above two learned feature maps using PrPool layers [21] and further mapped to feature vectors using fully-connected layers. Specifically, the output sizes of two PrPool layers are 8×8 and 4×4, respectively, and both following fully-connected layers output a 512-dimensional feature vector.
(4) Two 512-dimensional feature vectors of each RoI are concatenated to produce the learned feature vector of it. Its dimension, denoted as D, is 1024.
(5) The learned feature vectors of all training RoIs form the training data matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$. The test data matrix $\mathbf{Z} \in \mathbb{R}^{N\times D}$ is obtained in the same way.

It is worth noting that different from CFCF and CFNet whose training data matrices are circulant and most training samples are virtual ones, and different from DiMP whose training and test samples are always assumed to be square, in our DCFST, training data matrix is non-circulant and all training and test samples are real sampled identically to the actual size of the target object.

3.2 Discriminant Model Solver

We train a discriminant model that is differentiable and has a closed-form solution to fit the samples in $\mathbf{X}$ to their labels by integrating its solver into the proposed network. Because the discriminant model is differentiable and has a closed-form solution, its solver can be integrated into CNNs as a layer with forward and backward processes during training. As an instance, we apply the popular ridge regression model in this work to demonstrate the power of the proposed architecture. Ridge regression model has been confirmed to be simple, efficient and effective in the field of visual object tracking [24, 6, 40, 41, 48]. It can not only exploit all foreground and background samples to train a good regressor, but also effectively use high-dimensional features as the risk of over-fitting can be controlled by l2-norm regularization. Most importantly, it is differentiable and has a closed-form solution.

The optimization problem of ridge regression can be formulated as

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2, \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times D}$ contains D-dimensional feature vectors of N training samples, $\mathbf{y} \in \mathbb{R}^{N \times 1}$ is the vector containing their Gaussian labels, and $\lambda > 0$ is the regularization parameter. Its optimal solution can be expressed as

$$\mathbf{w}^* = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2}$$

Solving for $\mathbf{w}^*$ by directly using Eq. 2 is time-consuming because $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{D \times D}$ and the time complexity of matrix inversion is $O\left(D^3\right)$. Even if we obtain $\mathbf{w}^*$ by solving a system of linear equations with Gaussian elimination method, the time complexity is still $O\left(D^3/2\right)$, hindering the efficient training of the model when high-dimensional features are used. To address the dependence of the time complexity on the dimension of features, we employ the Woodbury formula [37]

$$\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{y}, \tag{3}$$

where $\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{N \times N}$. It is easy to see that the right hand of Eq. 3 allows us to solve for $\mathbf{w}^*$ in time complexity $O\left(N^3/2\right)$. Usually, the number of online training samples is smaller than the dimension of features in tracking, *i.e.*, $N < D$. Therefore, in order to solve for $\mathbf{w}^*$ efficiently, we use the right hand of Eq. 3 if the dimension of the learned feature vectors is larger than the number of training samples, *i.e*, $D > N$. Otherwise, the left hand is used.

Last but not least, when we integrate the ridge regression solver into the training of CNNs, it is necessary to calculate $\partial \mathbf{w}^* / \partial \mathbf{X}$ in the backward process.

Fortunately, $\partial \mathbf{w}^* / \partial \mathbf{X}$ is easy to be automatically obtained using the popular deep learning packages, such as TensorFlow and PyTorch, where automatic differentiation is supported. Specifically, during network training, given $\mathbf{X}$ and $\mathbf{y}$, only one line of code is necessary to solve for $\mathbf{w}^*$ in the forward process and there is no code needed in the backward process.

3.3 Fast Convergence with Shrinkage Loss

There exists extreme imbalance between foreground and background samples during network training. This problem slows down the convergence speed considerably and severely reduces the generalization ability of the learned feature embeddings if the commonly used mean square error loss $\mathcal{L}_{mse} = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$ is employed, where $\mathbf{y}$ and $\hat{\mathbf{y}} = \mathbf{Z}\mathbf{w}^*$ are vectors containing the ground-truth labels and the predicted labels of the N test samples in $\mathbf{Z}$, respectively. In fact, this is because most background samples are easy ones and only a few hard samples provide useful supervision, making the network training difficult.

To address this problem and make the network training efficient and effective, we propose a new shrinkage loss

$$\mathcal{L} = \left\| \frac{\exp(\mathbf{y}) \odot (\mathbf{y} - \hat{\mathbf{y}})}{1 + \exp(a \cdot (c - |\mathbf{y} - \hat{\mathbf{y}}|))} \right\|_2^2, \tag{4}$$

where a and c are hyper-parameters controlling the shrinkage speed and location, respectively, and the absolute value and the fraction are element-wise. Specifically, $\mathcal{L}$ down-weights the losses assigned to easy samples and mainly focuses on a few hard ones, preventing the vast number of easy backgrounds from overwhelming the learning of feature embeddings. It is easy-to-implement in a few lines of code using the current deep learning packages.

In fact, Eq. 4 is a modified version of the original shrinkage loss [33]

$$\mathcal{L}_o = \left\| \frac{\exp(\mathbf{y}) \odot (\mathbf{y} - \hat{\mathbf{y}})}{1 + \exp(a \cdot (c - (\mathbf{y} - \hat{\mathbf{y}})))} \right\|_2^2. \tag{5}$$

Mathematically, the main difference between Eq. 4 and Eq. 5 is that a sample is regarded as an easy one if its predicted value is larger than its label and their difference is less than c in Eq. 5, whereas in Eq. 4, we only consider the absolute difference to determine whether a sample is easy or not. In our experiments, we found that this modification can not only accelerate the convergence speed and reduce the validation loss in offline training, but also improve the accuracy in online tracking. Therefore, it may provide other researchers a better choice.

Moreover, it is worth noting that the motivations for using shrinkage loss in our approach and [33] are quite different. In [33], the purpose of using shrinkage loss is to prevent the discriminant model, *i.e.*, filters, from under-fitting to a few foreground samples after iterative training. However, there is no such concern in our approach because the discriminant model is trained directly by a close-form solution rather than an iterative method. The purpose of using shrinkage loss

in our approach is similar to that of using focal loss [32] in training detectors, that is, preventing the vast number of easy backgrounds from overwhelming the learning of feature embeddings.

Based on the above design and discussions, the resulting network can be trained in an end-to-end way, obtaining optimal feature embeddings for the discriminant model-based tracker. Ideally, in online tracking, the learned feature embeddings should make the corresponding discriminant model, *e.g.*, ridge regression model in this work, trained with the features extracted via them robust not only to the large appearance variation of the target object, but also to the heavy background clutters, thereby improving tracking accuracy.

4 Online Tracking with Learned Feature Embeddings

4.1 Features Extraction

Suppose $(\mathbf{p}_t, \mathbf{s}_t)$ denotes the position and size of the target object in frame t. Given frame t and $(\mathbf{p}_t, \mathbf{s}_t)$, we sample a square patch centered at $\mathbf{p}_t$, with an area of 5^2 times the target area, and then resize it to 288×288. Finally, the training data matrix $\mathbf{X}_t$ is obtained using the approach presented in Sec. 3.1. Similarly, given frame $t+1$ and $(\mathbf{p}_t, \mathbf{s}_t)$, the test data matrix $\mathbf{Z}_{t+1}$ can be obtained.

4.2 Online Learning and Update

In online tracking, according to Sec. 3.2, we can train a ridge regression model using the right hand or the left hand of Eq. 3 for online discriminative tracking. However, the time complexities of both sides are cubical with respect to D or N, hindering real-time performance. In order to solve for $\mathbf{w}^*$ more efficiently, we adopt the Gauss-Seidel based iterative approach [8]. Specifically, taking the left hand of Eq. 3 as an example, given $\mathbf{X}_t$ and $\mathbf{w}^*_{t-1}$, we decompose $\mathbf{X}_t^\top \mathbf{X}_t + \lambda \mathbf{I}$ into a lower triangular $\mathbf{L}_t$ and a strictly upper triangular $\mathbf{U}_t$, *i.e.*, $\mathbf{X}_t^\top \mathbf{X}_t + \lambda \mathbf{I} = \mathbf{L}_t + \mathbf{U}_t$. Then, $\mathbf{w}^*_t$ can be efficiently solved by iterative expressions:

$$\mathbf{w}_t^{*(j)} \leftarrow \mathbf{w}^*_{t-1}, \qquad j = 0, \tag{6a}$$

$$\mathbf{w}_t^{*(j)} \leftarrow \mathbf{L}_t \setminus \left(\mathbf{X}_t^\top \mathbf{y} - \mathbf{U}_t \mathbf{w}_t^{*(j-1)} \right), \qquad j > 0, \tag{6b}$$

where $\mathbf{w}^*_{t-1}$ is the trained model at frame $t-1$, and j indicates the number of iterations. In practice, 5 iterations are enough for the satisfactory $\mathbf{w}^*_t$. Note that this iterative method is efficient because Eq. 6b can be solved efficiently with forward substitution, and the time complexity of each iteration is $O\left(D^2\right)$ instead of $O\left(D^3/2\right)$.

In order to locate the target object robustly, updating the appearance model is necessary. Following the popular updating method used in [18, 24, 43, 48], we update $\mathbf{X}_t$ by means of the linear weighting approach, *i.e.*,

$$\begin{aligned} \widetilde{\mathbf{X}}_1 &= \mathbf{X}_1, \\ \widetilde{\mathbf{X}}_t &= (1-\delta)\, \widetilde{\mathbf{X}}_{t-1} + \delta \mathbf{X}_t, \qquad t > 1, \end{aligned} \tag{7}$$

where δ is the learning rate. As a result, instead of $\mathbf{X}_t$, $\widetilde{\mathbf{X}}_t$ is used in Eq. 6 for solving for $\mathbf{w}_t^*$. In addition, we keep the weight of $\mathbf{X}_1$ not being less than 0.25 during updating because the initial target information is always reliable.

4.3 Localization and Refine

Given the trained model $\mathbf{w}_t^*$ and test data matrix $\mathbf{Z}_{t+1}$, we locate the target by $\hat{\mathbf{y}}_{t+1} = \mathbf{Z}_{t+1}\mathbf{w}_t^*$, and the sample corresponding to the maximum element of $\hat{\mathbf{y}}_{t+1}$ is regarded as the target object. After locating the target, we refine its bounding box by ATOM [7] for more accurate tracking, similar to DiMP [4].

5 Experiments

Our DCFST is implemented in Python using PyTorch. On a single TITAN X(Pascal) GPU, employing ResNet-18 and ResNet-50 respectively as the backbone network, our DCFST-18 and DCFST-50 achieve tracking speeds of 35 FPS and 25 FPS, respectively. Code will be made available.

5.1 Implementation Details

Training Data. To increase the generalization ability of the learned feature embeddings, we use the training splits of recent large-scale tracking datasets, including TrackingNet [35], GOT10k [19], and LaSOT [11], in offline training. During network training, each pair of training and test images is sampled from a video snippet within the nearest 50 frames. For training image, we sample a square patch centered at the target, with an area of 5^2 times the target area. For test image, we sample a similar patch, with a random translation and scale relative to the target's. These cropped patches are then resized to a fixed size 288×288. We use image flipping and color jittering for data augmentation.
Training Setting. We use the pre-trained model on ImageNet to initialize the weights of our backbone network and freeze them during network training. The weights of our head network are randomly initialized with zero-mean Gaussian distributions. We train the head network for 40 epochs with 1.5k iterations per epoch and 48 pairs of images per batch, giving a total training time of 30(40) hours for DCFST-18(50) on a single GPU. The ADAM [26] optimizer is employed with initial learning rate of 0.005, using a factor 0.2 decay every 15 epochs.
Parameters. We sample 961 RoIs in each input image, *e.g.*, $N = 961$. The regularization parameter λ in Eq. 1 is set to 0.1. Two hyper-parameters a and c in Eq. 4 are set to 10 and 0.2, respectively. The learning rate δ in Eq. 7 is 0.01.

5.2 Ablation Studies

In this section, we will investigate the effect of choosing the loss function and hyper-parameter in our approach. Our ablation studies are based on DCFST-18 and performed on OTB2015 [45].

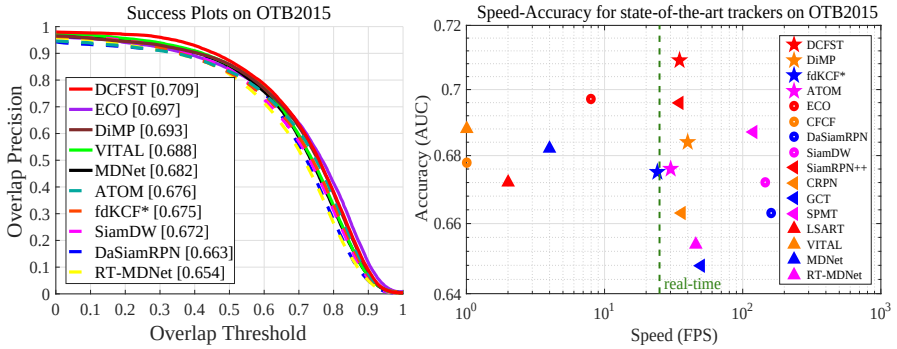
Table 1: Ablation studies on OTB2015. (a) The mean AUC scores of different loss functions. Our modified shrinkage loss achieves the best result. (b) The mean AUC scores and FPSs of different Ns. $N = 961$ achieves good balance.

(a) Loss Function.

Loss Function	Mean Square Error	Original Shrinkage	Modified Shrinkage
Mean AUC	0.682	0.698	0.709

(b) Number of Samples.

N	361 (19^2)	625 (25^2)	961 (31^2)	1369 (37^2)
Mean AUC	0.692	0.701	0.709	0.707
Mean FPS	46	40	35	32



(a) (b)

Fig. 2: State-of-the-art comparison on OTB2015. (a) The mean success plots of DCFST and nine state-of-the-art trackers. The mean AUC scores are reported in the legend. (b) Speed and accuracy plot of 16 state-of-the-art trackers. Our DCFST achieves the best accuracy, while running beyond the real-time speed.

Table 1a shows the mean AUC [45] scores of DCFST-18 with various loss functions. It can be seen that the tracking accuracy can be obviously improved (1.6% in AUC score) by using the original shrinkage loss instead of the mean square error one. This demonstrates that relaxing the imbalance of foreground-background samples in our approach is necessary and our choice of loss function is valid. Moreover, compared to the original shrinkage loss, our modified one can further improve the tracking accuracy by a large margin (1.1% in AUC score). This confirms the effectiveness of our improvement.

Table 1b shows the mean AUC scores and FPSs of DCFST-18 with various Ns. It can be seen that $N = 961$ can not only provide beyond the real-time speed, but also high tracking accuracy. It is worth mentioning that the stride of ResNet Block-3 is 8 pixel, and when $N = 31^2$ and the size of input image is 288×288, the interval between two adjacent samples is 7.68 pixel. Therefore, the tracking accuracy will not be improved significantly when $N > 961$.

Table 2: State-of-the-art comparison on OTB2015 in terms of mean overlap. The best three results are shown in red, blue, and magenta. Our DCFST outperforms its baseline tracker ATOM with a gain of 4%, using the same backbone network.

Tracker	DCFST	DiMP	fdKCF*	ATOM	SiamDW	DaSiamRPN	VITAL	ECO	MDNet	RT-MDNet
Mean Overlap	0.872	0.859	0.828	0.832	0.840	0.858	0.857	0.842	0.852	0.822

Table 3: State-of-the-art comparison on NFS in terms of mean AUC score. DiMP-18 and DiMP-50 are the DiMP tracker with ResNet-18 and ResNet-50 backbone network respectively. Both versions of our DCFST outperform other trackers.

Tracker	DCFST-50	DCFST-18	DiMP-50	DiMP-18	ATOM	UPDT	ECO	CCOT	MDNet	HDT	SFC
Mean AUC	0.641	0.634	0.620	0.610	0.584	0.537	0.466	0.488	0.429	0.403	0.401

Table 4: State-of-the-art comparison on the GOT10k test set in terms of average overlap (AO), and success rates (SR) at overlap thresholds 0.5 and 0.75. DaSiamRPN-18 is the DaSiamRPN tracker with ResNet-18 backbone network. Our DCFST-50 outperforms other trackers with large margins.

Tracker	DCFST-50	DCFST-18	DiMP-50	DiMP-18	ATOM	DaSiamRPN-18	CFNet	SiamFC	GOTURN
AO	0.638	0.610	0.611	0.579	0.556	0.483	0.374	0.348	0.347
SR(0.50)	0.753	0.716	0.717	0.672	0.634	0.581	0.404	0.353	0.375
SR(0.75)	0.498	0.463	0.492	0.446	0.402	0.270	0.144	0.098	0.124

Table 5: State-of-the-art comparison on the TrackingNet test set in terms of precision, normalized precision, and success. Our DCFST-50 achieves top results.

Tracker	DCFST-50	DCFST-18	DiMP-50	DiMP-18	ATOM	SiamRPN++	CRPN	SPMT	DaSiamRPN	CFNet
Precision	0.700	0.682	0.687	0.666	0.648	0.694	0.619	0.661	0.591	0.533
Norm. Prec.	0.809	0.797	0.801	0.785	0.771	0.800	0.746	0.778	0.733	0.654
Success (AUC)	0.752	0.739	0.740	0.723	0.703	0.733	0.669	0.712	0.638	0.578

5.3 State-of-the-art Comparisons

OTB2015 [45]. OTB2015 is the most popular benchmark for the evaluation of trackers, containing 100 videos with various challenges. On the OTB2015 experiment, we first compare our DCFST-18 against nine state-of-the-art trackers, DiMP [4], fdKCF* [48], ATOM [7], SiamDW [46], DaSiamRPN [51], VITAL [39], ECO [6], MDNet [36], and RT-MDNet [22]. Success plot, mean overlap precision and AUC score [45] are employed to quantitatively evaluate all trackers. Fig. 2a

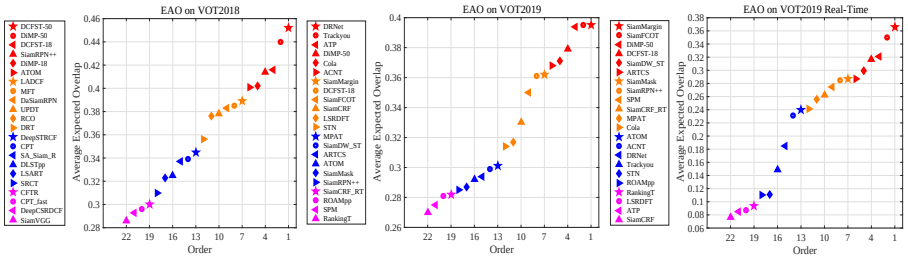


Fig. 3: State-of-the-art comparison on VOT2018, VOT2019, and VOT2019 real-time challenges in terms of expected average overlap (EAO).

and Table 2 show the results. Our DCFST-18 obtains the mean AUC score and overlap precision of 70.9% and 87.2%, outperforming the second best trackers (ECO and DiMP) with significant gains of 1.2% and 1.3%, respectively. Additionally, Fig. 2b shows the comparison of DCFST-18 with the above trackers along with SiamRPN++ [30], CRPN [12], GCT [14], SPMT [44], LSART [40], and CFCF [15] in both mean AUC score and FPS. Our DCFST-18 achieves the best trade-off between accuracy and speed among all state-of-the-art trackers.

NFS [23]. We evaluate our DCFST on the 30 FPS version of NFS benchmark which contains 100 challenging videos with fast-moving objects. On the NFS experiment, we compare DCFST against DiMP, ATOM, UPDT [5], ECO, CCOT [9] along with the top-3 trackers, MDNet, HDT [38], and SFC [3], evaluated by NFS. All trackers are quantitatively evaluated by AUC score. Table 3 shows the results. Our DCFST-18 and DCFST-50 obtain the mean AUC scores of 63.4% and 64.1%, outperforming the latest state-of-the-art trackers DiMP-18 and DiMP-50 with gains of 2.4% and 2.1%, respectively. Additionally, DCFST-18 surpasses its baseline tracker ATOM [6] with a significant gain of 5.1%.

GOT10k [19]. We evaluate our DCFST on the test set of GOT10k which is a large-scale tracking benchmark and contains over 9000 training videos and 180 test videos. Here, the generalization capabilities of the tracker to unseen object classes is of major importance. Therefore, to ensure a fair comparision, on the GOT10k experiment, we retrain our DCFST and then compare it against DiMP, ATOM, DaSiamRPN along with the top-3 trackers, CFNet [43], SiamFC [2], and GOTURN [17], evaluated by GOT10k. Following the GOT10k challenge protocol, all trackers are quantitatively evaluated by average overlap, and success rates at overlap thresholds 0.5 and 0.75. Table 4 shows the results. In terms of success rate at overlap thresholds 0.5, our DCFST-18 and DCFST-50 obtain the scores of 71.6% and 75.3%, outperforming the latest state-of-the-art trackers DiMP-18 and DiMP-50 with gains of 4.4% and 3.6%, respectively. Additionally, DCFST-18 surpasses its baseline tracker ATOM with a significant gain of 8.2%.

TrackingNet [35]. We evaluate our DCFST on the test set of TrackingNet which is a large-scale tracking benchmark and provides 511 test videos to as-

[6] We state the relationship between DCFST and ATOM in supplementary materials.

sess trackers. On the TrackingNet experiment, we compare DCFST against seven state-of-the-art trackers, DiMP, ATOM, SiamRPN++, DaSiamRPN, SPMT, CRPN, and CFNet. Following the TrackingNet challenge protocol, all trackers are quantitatively evaluated by precision, normalized precision, and AUC score. Table 5 shows the results. Our DCFST-18 and DCFST-50 obtain the mean AUC scores of 73.9% and 75.2%, outperforming the latest state-of-the-art trackers DiMP-18 and DiMP-50 with gains of 1.6% and 1.2%, respectively. Additionally, DCFST-18 surpasses its baseline tracker ATOM with a significant gain of 3.6%.
VOT2018/2019 [27, 28]. We evaluate our DCFST on the 2018 and 2019 versions of Visual Object Tracking (VOT) challenge which contain 60 sequences, respectively. On the VOT2018 experiment, we compare DCFST against SiamRPN++, DiMP, ATOM along with the top-16 trackers on VOT2018 challenge. On the VOT2019 experiment, we compare DCFST-18 against the top-21 trackers on VOT2019 and VOT2019 real-time challenges, respectively. Following the VOT challenge protocol, all trackers are quantitatively evaluated by expected average overlap (EAO). Fig. 3 shows the results. (1) On the VOT2018 challenge, our DCFST-18 and DCFST-50 obtain the EAO scores of 0.416 and 0.452, outperforming the latest state-of-the-art trackers DiMP-18 and DiMP-50 with gains of 1.4% and 1.2%, respectively. Additionally, DCFST-50 outperforms all other state-of-the-art trackers including SiamRPN++, and DCFST-18 surpasses its baseline tracker ATOM with a significant gain of 1.5%. (2) On the VOT2019 and VOT2019 real-time challenges, our DCFST-18 achieves the EAO scores of 0.361 and 0.317, respectively, surpassing its baseline tracker ATOM with significant gains of 6.9% and 7.7%, respectively. There are seven latest trackers perform well than DCFST-18 on VOT2019 challenge, however, five of them are obviously lower than DCFST-18 in terms of real-time performance. Although SiamRPN++ employs stronger backbone network (ResNet-50) and more training datas (ImageNet DET, ImageNet VID, YouTube, and COCO) than DCFST-18, DCFST-18 consistently outperforms it on all three challenges with large margins.

6 Conclusion

A novel and state-of-the-art tracker DCFST is proposed in this paper. By integrating the solver of a discriminant model that is differentiable and has a closed-form solution into convolutional neural networks, DCFST learns optimal feature embeddings for the discriminant model-based tracker in offline training, thus improves its accuracy and robustness in online tracking. Extensive experiments on multiple challenging benchmarks show that DCFST achieves state-of-the-art accuracy and beyond the real-time speed, and also sets a simple yet strong baseline for visual tracking due to its simplicity, efficiency and effectiveness.
Acknowledgements. This work was supported by the Research and Development Projects in the Key Areas of Guangdong Province (No. 2020B010165001). This work was also supported by National Natural Science Foundation of China under Grants 61772527, 61976210, 61806200, 61702510 and 61876086.

References

1. Bertinetto, L., Henriques, J.F., Torr, P., Vedaldi, A.: Meta-learning with differentiable closed-form solvers. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=HyxnZh0ct7
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European conference on computer vision. pp. 850–865. Springer (2016)
3. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: European conference on computer vision. pp. 850–865. Springer (2016)
4. Bhat, G., Danelljan, M., Gool, L.V., Timofte, R.: Learning discriminative model prediction for tracking. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
5. Bhat, G., Johnander, J., Danelljan, M., Shahbaz Khan, F., Felsberg, M.: Unveiling the power of deep tracking. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 483–498 (2018)
6. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Eco: efficient convolution operators for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6638–6646 (2017)
7. Danelljan, M., Bhat, G., Shahbaz Khan, F., Felsberg, M.: Atom: Accurate tracking by overlap maximization. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
8. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: Proceedings of the IEEE international conference on computer vision. pp. 4310–4318 (2015)
9. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision. pp. 472–488. Springer (2016)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
11. Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., Ling, H.: Lasot: A high-quality benchmark for large-scale single object tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5374–5383 (2019)
12. Fan, H., Ling, H.: Siamese cascaded region proposal networks for real-time visual tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
13. Gao, J., Zhang, T., Xu, C.: Graph convolutional tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
14. Gao, J., Zhang, T., Xu, C.: Graph convolutional tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
15. Gundogdu, E., Alatan, A.A.: Good features to correlate for visual tracking. IEEE Transactions on Image Processing **27**(5), 2526–2540 (2018)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Held, D., Thrun, S., Savarese, S.: Learning to track at 100 fps with deep regression networks. In: European Conference Computer Vision (ECCV) (2016)

18. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE transactions on pattern analysis and machine intelligence **37**(3), 583–596 (2014)
19. Huang, L., Zhao, X., Huang, K.: Got-10k: A large high-diversity benchmark for generic object tracking in the wild. arXiv preprint arXiv:1810.11981 (2018)
20. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
21. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 784–799 (2018)
22. Jung, I., Son, J., Baek, M., Han, B.: Real-time mdnet. In: European Conference on Computer Vision (ECCV) (Sept 2018)
23. Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., Lucey, S.: Need for speed: A benchmark for higher frame rate object tracking. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1125–1134 (2017)
24. Kiani Galoogahi, H., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1135–1143 (2017)
25. Kiani Galoogahi, H., Sim, T., Lucey, S.: Correlation filters with limited boundaries. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4630–4638 (2015)
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
27. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Cehovin Zajc, L., Vojir, T., Bhat, G., Lukezic, A., Eldesokey, A., et al.: The sixth visual object tracking vot2018 challenge results. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 0–0 (2018)
28. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Pflugfelder, R., Kamarainen, J.K., Čehovin Zajc, L., Drbohlav, O., Lukezic, A., Berg, A., Eldesokey, A., Kapyla, J., Fernandez, G.: The seventh visual object tracking vot2019 challenge results (2019)
29. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
30. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
31. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8971–8980 (2018)
32. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
33. Lu, X., Ma, C., Ni, B., Yang, X., Reid, I., Yang, M.H.: Deep regression tracking with shrinkage loss. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 353–369 (2018)
34. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: Proceedings of the IEEE international conference on computer vision. pp. 3074–3082 (2015)

35. Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., Ghanem, B.: Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 300–317 (2018)
36. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4293–4302 (2016)
37. Petersen, K.B., Pedersen, M.S., et al.: The matrix cookbook. Technical University of Denmark **7**(15), 510 (2008)
38. Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.H.: Hedged deep tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4303–4311 (2016)
39. Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., Shen, C., Lau, R.W., Yang, M.H.: Vital: Visual tracking via adversarial learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8990–8999 (2018)
40. Sun, C., Wang, D., Lu, H., Yang, M.H.: Learning spatial-aware regressions for visual tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8962–8970 (2018)
41. Tang, M., Yu, B., Zhang, F., Wang, J.: High-speed tracking with multi-kernel correlation filters. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
42. Tao, R., Gavves, E., Smeulders, A.W.: Siamese instance search for tracking. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1420–1429 (2016)
43. Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., Torr, P.H.: End-to-end representation learning for correlation filter based tracking. In: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. pp. 5000–5008. IEEE (2017)
44. Wang, G., Luo, C., Xiong, Z., Zeng, W.: Spm-tracker: Series-parallel matching for real-time visual object tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
45. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1834–1848 (2015)
46. Zhang, Z., Peng, H.: Deeper and wider siamese networks for real-time visual tracking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
47. Zheng, L., Chen, Y., Tang, M., Wang, J., Lu, H.: Siamese deformable cross-correlation network for real-time visual tracking. Neurocomputing **401**, 36–47 (2020)
48. Zheng, L., Tang, M., Chen, Y., Wang, J., Lu, H.: Fast-deepkcf without boundary effect. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4020–4029 (2019)
49. Zheng, L., Tang, M., Chen, Y., Wang, J., Lu, H.: High-speed and accurate scale estimation for visual tracking with gaussian process regression. In: 2020 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2020)
50. Zheng, L., Tang, M., Wang, J.: Learning robust gaussian process regression for visual tracking. In: IJCAI. pp. 1219–1225 (2018)
51. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 101–117 (2018)