Real-Time Correlation Tracking Via Joint Model Compression and Transfer

Ning Wang, Wengang Zhou, *Member, IEEE*, Yibing Song, Chao Ma, *Member, IEEE*, and Houqiang Li, *Senior Member, IEEE*

***Abstract*— Correlation filters (CF) have received considerable attention in visual tracking because of their computational efficiency. Leveraging deep features via off-the-shelf CNN models (e.g., VGG), CF trackers achieve state-of-the-art performance while consuming a large number of computing resources. This limits deep CF trackers to be deployed to many mobile platforms on which only a single-core CPU is available. In this paper, we propose to jointly compress and transfer off-the-shelf CNN models within a knowledge distillation framework. We formulate a CNN model pretrained from the image classification task as a teacher network, and distill this teacher network into a lightweight student network as the feature extractor to speed up CF trackers. In the distillation process, we propose a fidelity loss to enable the student network to maintain the representation capability of the teacher network. Meanwhile, we design a tracking loss to adapt the objective of the student network from object recognition to visual tracking. The distillation process is performed offline on multiple layers and adaptively updates the student network using a background-aware online learning scheme. The online adaptation stage exploits the background contents to improve the feature discrimination of the student network. Extensive experiments on six standard datasets demonstrate that the lightweight student network accelerates the speed of state-of-the-art deep CF trackers to real-time on a single-core CPU while maintaining almost the same tracking accuracy.**

***Index Terms*— Correlation tracking, model transfer, knowledge distillation, real-time tracking.**

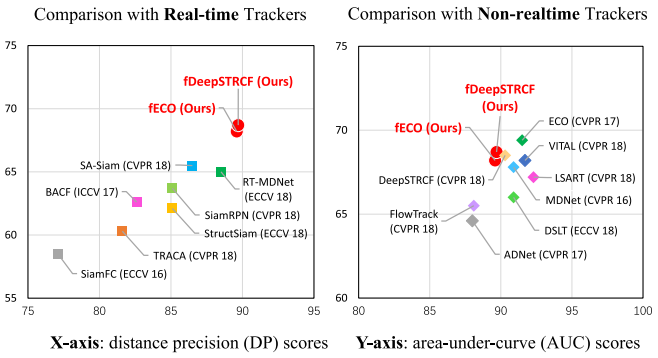


Fig. 1. Tracking results on the OTB-2015 dataset [10]. The proposed method accelerates state-of-the-art deep CF trackers (i.e., ECO [7] and DeepSTRCF [11]) through joint CNN model compression and transfer. The improved CF trackers (i.e., fECO and fDeepSTRCF) perform favorably against existing methods and achieve real-time or near real-time speeds on a single-core CPU. It is worth mentioning that most existing real-time trackers as shown on the left cannot achieve real-time speed on a CPU, while the recent performance leaders shown on the right are far from real-time even on a GPU.

I. Introduction

THERE has been an increasing demand for visual object tracking algorithms in numerous vision applications. Typical examples include video surveillance, human-computer interaction, and autonomous driving [1], [2]. As a key component, tracking target objects in real-time plays a critical role in improving the overall efficiency of vision applications. The visual tracking framework based on Correlation Filters (CF) has been widely investigated in [3]–[5] because of the efficient correlation computation in the Fourier domain. When integrated with CNN features, CF trackers [6]–[8] achieve state-of-the-art tracking accuracy. However, extracting high-dimensional deep features brings in a huge computational cost and limits CF trackers to achieve real-time speed. Although deep operations can always be accelerated by GPUs, deep CF trackers cannot be deployed on CPU-only devices, e.g., most intelligent mobile phones do not have GPUs, let alone the huge power consumption and memory storage required by existing pretrained CNN models (e.g., VGG [9]). The challenges of using off-the-shelf CNN models (e.g., VGG [9]) include huge demand for memory storage, heavy computational burden, and high power consumption. It is therefore non-trivial to investigate how to accelerate deep CF trackers on a CPU platform to achieve real-time speed without suffering a significant drop in tracking accuracy.

In this paper, we jointly compress and transfer off-the-shelf CNN models into a lightweight feature extractor. The lightweight feature extractor enables deep CF trackers to

Manuscript received July 23, 2019; revised February 15, 2020; accepted April 13, 2020. Date of publication April 28, 2020; date of current version May 5, 2020. The work of Wengang Zhou was supported in part by the NSFC under Contract 61822208 and Contract 61632019 and in part by the Youth Innovation Promotion Association CAS under Grant 2018497. The work of Chao Ma was supported in part by the NSFC under Grant 61906119 and in part by the Shanghai Pujiang Program. The work of Houqiang Li was supported in part by the NSFC under Contract 61836011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Adrian Munteanu. *(Corresponding authors: Wengang Zhou; Houqiang Li.)*

Ning Wang, Wengang Zhou, and Houqiang Li are with the CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System, Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230026, China (e-mail: wn6149@mail.ustc.edu.cn; zhwg@ustc.edu.cn; lihq@ustc.edu.cn).

Yibing Song is with the Tencent AI Lab, Shenzhen 518000, China (e-mail: dynamicstevenson@gmail.com).

Chao Ma is with the MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: chaoma@sjtu.edu.cn).

Digital Object Identifier 10.1109/TIP.2020.2989544

achieve real-time speed as well as consume less memory. The model compression and transfer are from the perspective of knowledge distillation [12], [13]. We take the off-the-shelf model as a teacher network, which is pretrained for the object recognition task. On the other side, a low capacity student network is used to learn from the teacher network. In the distillation process, we propose two types of losses. The first one is a fidelity loss and the second one is a correlation tracking loss. The fidelity loss ensures the student network to convey the representation from the teacher network, while the correlation tracking loss transfers the objective of the student network from object recognition to visual tracking. We take the hierarchies of deep models into account and perform the distillation process on multiple CNN layers offline. After distillation, the student network maintains the high-level semantic discrimination from the fidelity loss. Besides, the tracking loss helps the student network to produce target-specific CNN representations. During online tracking, we propose a background-aware adaptation method to update the student network for further performance improvement.

The student network is a lightweight feature extraction backbone. The model size of the student network is only 1.5 MB while the original size of the teacher network is 95 MB (i.e., 63$\times$ smaller). When integrated with the proposed lightweight backbone, the state-of-the-art deep CF trackers including ECO [7] and DeepSTRCF [11] are able to achieve real-time speed on a single-core CPU while maintaining almost the same tracking accuracy on prevalent tracking benchmarks.

We summarize the contributions of our work as follows:

- We compress and adapt off-the-shelf deep CNN models into lightweight backbones by knowledge distillation. We propose a fidelity loss and a correlation tracking loss to jointly compress the network and transfer its objective from object recognition to visual tracking.
- We propose to distillate student network via hierarchical CNN representations offline. We propose a background-aware adaption method to incrementally fine-tune the student network to adapt to target appearance changes.
- We integrate the proposed lightweight backbone into the state-of-the-art deep CF trackers [7], [11]. Evaluations on the large-scale benchmark datasets indicate the effectiveness of the proposed method in terms of the real-time speed and tracking accuracy.

In the following of the paper, we describe the related work in Section II, correlation tracking in Section III, the proposed approach in Section IV, and experiments in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

In this section, we briefly survey the closely related literature on three aspects: tracking by correlation filters, real-time tracking, and network compression.

A. Correlation Tracking

Correlation filters have been widely studied in visual tracking since the MOSSE method [3] was proposed by Bolme *et al.* in 2010. The correlation filter is trained by minimizing a ridge regression loss for all circular shifts of the training sample, which can be efficiently solved in the Fourier domain [14]. Heriques *et al.* exploited the circulant structure of training patches in the kernel space [4]. The SRDCF tracker [15] alleviates the boundary effects by penalizing correlation filter coefficients depending on spatial locations. The CSR-DCF algorithm [16] constructs filters with channel and spatial reliability. The C-COT [17] adopts a continuous-domain formulation and is further improved by an efficient convolution operator (ECO [7]). The recent DRT tracker [18] jointly learns the discrimination and reliability of CF. In addition, multiple kernels [19], combination with particle filter [20], re-detection mechanism for long-term scenario [21], [22] and ensemble learning schemes [23]–[25] have also been investigated in the CF family. In recent years, the combination of CF trackers and deep features from off-the-shelf CNN models has demonstrated impressive results [6], [7], [26]. Even though state-of-the-art results can be obtained by leveraging deep feature representations, the characteristic real-time efficiency of the correlation filter has gradually faded due to the adopted heavyweight CNN model. In this work, different from the above approaches putting emphasis on learning more discriminative filters, we focus on learning a distilled lightweight backbone network that enables high-performance real-time correlation tracking even on a single-core CPU.

B. Real-Time Tracking

As the basic component in practical vision applications, tracking algorithms generally require the high efficiency to handle video frames in real-time. The recent real-time trackers can be roughly categorized into Siamese tracking and CF based tracking. The Siamese network [27] regards the tracking task as a similarity learning problem, and compares the template patch with the candidate patches in the search patch in a sliding-window manner. On the basis of the SiamFC framework [27], the correlation layer [28], attention mechanism [29], semantic branch [30] and unsupervised learning scheme [31] are widely explored. The recent region proposal Siamese network [32], [33] achieves higher speed compared with SiamFC [27] by discarding multiple-scale estimation. However, the Siamese networks heavily rely on powerful GPUs and the running speed on CPU is only 2$\sim$3 FPS [34] due to heavyweight model complexity.

On the other hand, CF trackers can achieve real-time speed when using lightweight hand-crafted features such as HOG and ColorNames [4], [11], [24], [35], but they typically have an obvious performance gap with the remarkable deep CF trackers. Equipped with CNN features, CF trackers achieve state-of-the-art tracking accuracy but suffer from a large computational cost. Methods of feature dimension reduction, such as PCA [36], factorized convolution operator [7], and encoder network [37], can reduce the feature complexity to some extent. However, these methods have to first extract high-dimensional CNN features from the original heavyweight deep models. In contrast, our method produces a lightweight network offline for efficient feature extraction, which not only

naturally reduces the feature dimension but also greatly saves the feature extraction time.

C. Network Compression

There are two typical network compression approaches involving model pruning and knowledge distillation. Model pruning [38] usually removes unimportant filter weights and utilizes online fine-tuning to recover accuracy. Knowledge distillation [12], [13] is based on the observation that a small network has similar representation capability as a large network but is usually harder to train solely [39]. Knowledge distillation [12], [13] uses a powerful teacher network to guide a smaller student network. The student is forced to mimic the feature representation [13], [39] or classification probabilities [12] of its teacher. However, previous methods usually compress models directly on the same vision task (e.g., image classification). In contrast, our method not only compresses the deep models but also transfers the objective to the tracking task. Therefore, the distillation and tracking processes are jointly optimized in an end-to-end manner. Unlike existing methods that usually compress the model by $4\times$ or $8\times$ with limited speed acceleration [12], [13], by virtue of collaborative training, we achieve a much larger compression rate of $63\times$ while maintaining almost the same tracking accuracy. In [40], the classic knowledge distillation scheme is used to compress off-the-shelf CNN networks in the tracking framework. We note that how to bridge the gap between object recognition and visual tracking is not fully explored. In this work, we propose to simultaneously distill the pretrained networks and narrow the task gap, which helps our method to achieve a much higher compression rate and a real-time speed on CPU. To further reduce the model degradation caused by compression, we propose multiple-level knowledge transfer and employ a background-aware online adaption scheme to fine-tune the student network for each sequence.

III. Revisiting Correlation Tracking

A typical CF based tracker [3], [4] is trained using an image patch $\mathbf{x}$ centered around the target. All of the circular shifts of the target patch $\mathbf{x}$ are generated as training samples with Gaussian function labels. Considering the feature embedding $\varphi(\cdot)$, the filter $\mathbf{w}$ can be trained by minimizing the following regularized regression objective:

$$\min_{\mathbf{w}} \left\| \sum_{i=1}^{D} \varphi_i(\mathbf{x}) \star \mathbf{w}_i - \mathbf{y} \right\|^2 + \lambda \sum_{i=1}^{D} \|\mathbf{w}_i\|^2, \tag{1}$$

where λ is a regularization parameter, D is the number of feature channel, $\star$ denotes the circular correlation and $\mathbf{y}$ is the desired Gaussian label. The correlation filter on the d-th ($d \in \{1, \cdots, D\}$) channel can be efficiently learned as follows:

$$\hat{\mathbf{w}}_d = \frac{\hat{\mathbf{y}}^* \odot \hat{\varphi}_d(\mathbf{x})}{\sum_{i=1}^{D} \hat{\varphi}_i(\mathbf{x}) \odot \hat{\varphi}_i^*(\mathbf{x}) + \lambda}, \tag{2}$$

where $\odot$ is the element-wise product, hat notation $\hat{\cdot}$ denotes the Discrete Fourier Transform (DFT) and $\cdot^*$ is the complex-conjugate operation.

In the next frame, a search patch $\mathbf{z}$ with the same size of $\mathbf{x}$ is cropped out for predicting the target position, and the corresponding response $\mathbf{r}$ is computed by

$$\mathbf{r} = \mathcal{F}^{-1}\left(\sum_{i=1}^{D} \hat{\mathbf{w}}_i^* \odot \hat{\varphi}_i(\mathbf{z}) \right), \tag{3}$$

where $\mathcal{F}^{-1}(\cdot)$ is the inverse DFT. Since a higher feature dimension D implies a larger computation burden, a lightweight feature backbone network not only accelerates feature extraction but also expedites the correlation filter learning (Eq. (2)) and detection (Eq. (3)) processes.

In this work, we aim to train a lightweight backbone network for efficient correlation tracking. To verify the effectiveness and generality, we select a baseline and two state-of-the-art CF frameworks as follows:

- KCF [4] is a plain CF tracker. We use it to verify the feature representation capability between the teacher and student networks.
- ECO [7] is based on the C-COT [17] tracker and integrates several efficient strategies. ECO adopts the features from VGG-M. We develop the fast version (fECO) using our distilled lightweight model.
- STRCF [11] is a CF tracker with a Spatial-Temporal Regularization. STRCF shows impressive performance with hand-crafted features, and DeepSTRCF using VGG-M achieves further improvement but the speed is greatly limited. We implement a fast version, namely fDeepSTRCF, using our model.

IV. Proposed Method

Figure 2 shows an overview of our framework involving offline knowledge distillation and online prediction. In the offline knowledge distillation step, we use two teacher networks and two shared-weight student networks. The VGG-M [41] is selected as the teacher network, which is widely used in deep CF trackers [7], [11], [17], [18], [42]. The teacher network is frozen without gradient back-propagation in the training stage. We first randomly prune the teacher network to initialize the student network. Specifically, for one convolutional layer of the teacher network, we randomly prune 7/8 filters in the current layer and the corresponding 7/8 channels in each filter of the next convolutional layer. The student networks aim to produce similar feature representations of the teacher networks while reducing around 63 times of the model storage. Figure 3 shows the detailed architectures of the student and teacher networks where the filter capacity of the student network is 64 times smaller than that of the teacher network in each layer except in the first layer. As a result, the teacher network without fully connected layers is 95 MB while our lightweight model is only 1.5 MB. We denote the distilled student network as CF-VGG.

In the following, we first introduce how to offline compress and transfer deep models for efficient tracking in subsection IV-A. Then we present the efficient online adaptation scheme in subsection IV-B.

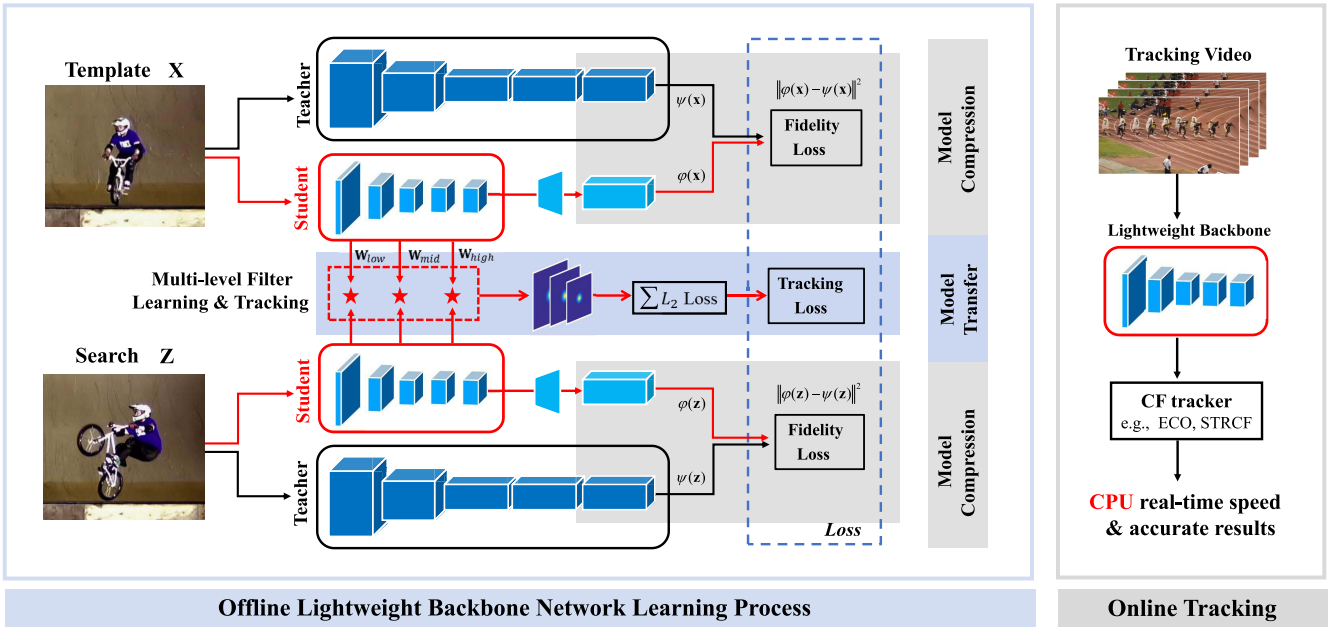


Fig. 2. Pipeline of knowledge distillation and online prediction. We learn to offline compress the teacher network by using the proposed fidelity loss and correlation tracking loss. For model transfer, we take the first, second and last convolutional layers before the pooling operations as the low, middle and high-level representations, respectively. In the online stage, the distilled student network adapts to the target object of each input video sequence and helps track the target object real-time on a single-core CPU.

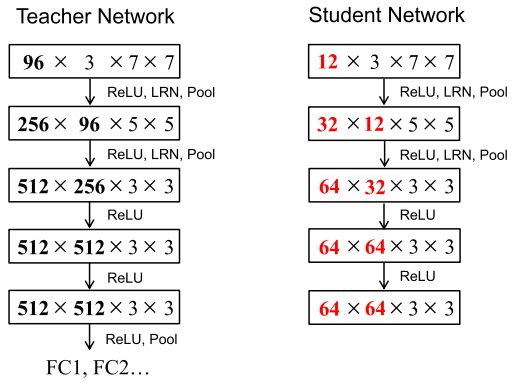


Fig. 3. Architecture comparison between the teacher and student networks. The numbers in each convolutional layer indicate the filter number, filter channel, filter width and height, respectively. Notice that the student network reduces 8 times of both filter numbers and channels, which takes around 64 times smaller than the teacher network.

A. Joint Model Transfer and Compression

In the offline training step, we propose two types of losses to simultaneously compress and transfer the teacher network: 1) The fidelity loss ensures the same feature representation capability between the student and teacher networks. 2) The correlation tracking loss transfers the source objective of classification into the target objective of regression for tracking. The fidelity loss mainly maintains the semantic description in high levels, while the tracking loss learns the similarity (or template matching) to evaluate the minor appearance changes of target objects between frames. By joint training, semantic features can complement the appearance features. These two losses constitute the final objective function, which is formulated as:

$$\mathcal{L}_{\text{offline}} = \mathcal{L}_{\text{tracking}} + \lambda \mathcal{L}_{\text{fidelity}} + \gamma \|\Theta\|^2, \tag{4}$$

where λ is a hyper-parameter balancing the influences of these two losses, Θ denotes the learnable parameters of the student network and the last term is the weight decay. In the following, we present the details of the semantic fidelity loss $\mathcal{L}_{\text{fidelity}}$ and the correlation tracking loss $\mathcal{L}_{\text{tracking}}$.

1) Semantic Fidelity Loss: Once we initialize the student network by filter pruning, the feature dimensions of the student and teacher networks are different. We use a 1×1 fully convolutional operation to match their feature dimension. Given a target patch $\mathbf{x}$ and a search patch $\mathbf{z}$, the features from the student network and the teacher network should be as similar as possible. We propose a fidelity loss to measure the feature differences. Formally, we define the fidelity loss as:

$$\begin{aligned}\mathcal{L}_{\text{fidelity}} &= \mathcal{L}_{\text{target}} + \mathcal{L}_{\text{search}} \\ &= \|\varphi(\mathbf{x}) - \psi(\mathbf{x})\|^2 + \|\varphi(\mathbf{z}) - \psi(\mathbf{z})\|^2,\end{aligned} \tag{5}$$

where $\varphi(\cdot)$ represents the trainable feature embedding of the student network (its notation Θ is omitted for clarity), and $\psi(\cdot)$ is the fixed embedding of the teacher network.

2) Correlation Tracking Loss: In addition to the fidelity loss, we propose the correlation tracking loss to modify the objective of the student network from classification to regression. We feed the search patch and the target patch into the student network to obtain their features and use a CF to model the response map regression. The circular correlation can be computed in the Fourier domain with a closed-form

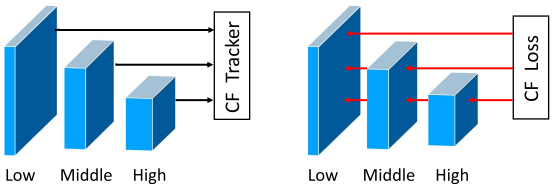


Fig. 4. Existing deep CF trackers learn correlation filters directly on multi-layer CNN features (left). Our framework aims at distilling a lightweight CNN backbone using back propagation to fine-tune multiple layers (right).

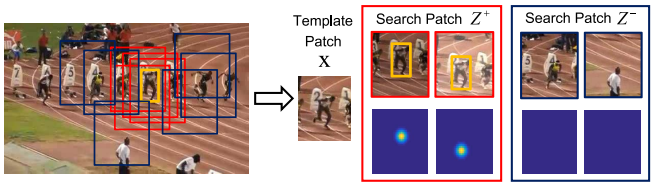


Fig. 5. Illustration of sample generation for background-aware online adaptation. Given the first frame, the template $\mathbf{x}$ is cropped centered at the target position. The foreground patches $\mathbf{z}^+$ contain the target. We augment the foreground patches for training. The background patches $\mathbf{z}^-$ do not include the target and their corresponding labels are set to zero.

solution [4], [35] and the backward formulas can also be efficiently derived. The corresponding loss function is the L_2 distance between the correlation response map $\mathbf{r}$ and the ground truth label $\mathbf{g}$ as follows:

$$\begin{aligned} \mathcal{L}_{\text{tracking}} &= \|\mathbf{r}-\mathbf{g}\|^2, \\ s.t. \quad \mathbf{r} &= \mathcal{F}^{-1}\left(\hat{\mathbf{w}}^* \odot \hat{\varphi}(\mathbf{z})\right), \\ \hat{\mathbf{w}} &= \frac{\hat{\mathbf{y}}^* \odot \hat{\varphi}(\mathbf{x})}{\hat{\varphi}(\mathbf{x}) \odot \hat{\varphi}^*(\mathbf{x}) + \lambda}, \end{aligned} \tag{6}$$

where $\mathbf{g}$ is a Gaussian map centered at the annotated target location. For clarity, in comparison with Eq. (2), we omit the feature dimension D in Eq. (6) and the subsequent equations. The back-propagation of the above loss with respect to $\varphi(\mathbf{x})$ and $\varphi(\mathbf{z})$ are given by Eq. (7) below. Interested readers can refer to [28], [43] for more details.

$$\begin{aligned} \nabla_{\varphi(\mathbf{x})}\mathcal{L} &= \mathcal{F}^{-1}\left(\nabla_{\hat{\varphi}^*(\mathbf{x})}\mathcal{L} + (\nabla_{\hat{\varphi}(\mathbf{x})}\mathcal{L})^*\right), \\ \nabla_{\varphi(\mathbf{z})}\mathcal{L} &= \mathcal{F}^{-1}\left(\nabla_{\hat{\varphi}^*(\mathbf{z})}\mathcal{L}\right). \end{aligned} \tag{7}$$

Furthermore, we perform the model transfer with multiple-level feature representations. Unlike existing deep CF trackers [6], [7], [17], [26] that simply integrate multiple CNN layers with empirical or learnable weights to boost the performance (see Figure 4), we separately apply the trainable constraint to multiple CNN layers to fine-tune the student network. This helps the student network not only fit the correlation tracking task better but also maintain a richer representation capability than only using the features from the last CNN layer (see more experiments in subsection V-B). In this work, we take the first, second and last convolutional layers before their pooling operations as the low, middle and high-level feature representations, respectively. The final tracking loss is formulated as:

$$\begin{aligned} \mathcal{L}_{\text{tracking}} &= \sum_l \|\mathbf{r}_l - \mathbf{g}_l\|^2, \\ s.t. \quad \mathbf{r}_l &= \mathcal{F}^{-1}\left(\hat{\mathbf{w}}_l^* \odot \hat{\varphi}_l(\mathbf{z})\right), \\ l &\in \{\text{high, middle, low}\}, \end{aligned} \tag{8}$$

where l means the index of the feature representation level. $\mathbf{g}_l$ contains the ground truth labels, which are all Gaussian maps but with different spatial sizes. $\varphi_l(\cdot)$ denotes the feature embedding of the student network on the l-th level. The CFs with different levels of features (i.e., $\mathbf{w}_l$) are learned using Eq. (2).

B. Background-Aware Online Adaptation

The offline distillation decreases the network capacity while preserving the feature representation. In the tracking scenarios, objects belonging to the same category may be labeled differently according to the first frame annotations. Figure 5 shows an example where only one athlete is positively labeled while the remaining are labeled as negative. In order to increase the feature discrimination, we online fine-tune the student network using the annotations in the first frame. Our idea is motivated by the context-aware correlation filter (CACF) [44] that regresses hard negative samples $\mathbf{x}^-$ to the negative labels. These hard negative samples do not overlap with the target object. In CACF [44], the context-aware information is learned through:

$$\min_{\mathbf{w}} \|\varphi(\mathbf{x}^+) \star \mathbf{w} - \mathbf{y}\|^2 + \lambda_1\|\mathbf{w}\|^2 + \lambda_2 \sum_{i=1}^{k} \|\varphi(\mathbf{x}_i^-) \star \mathbf{w}\|^2, \tag{9}$$

where $\mathbf{x}^+$ is the positive training sample including the target and $\mathbf{x}_i^-$ collects the negative samples that do not overlap with the target region. Given pretrained deep features, the CACF method enhances the filter-level discriminative capability. However, in our work, we explore the background-aware information in model training to boost the feature-level representation.

In the offline distillation step, all the training samples contain the target object, which helps discriminate the target from the background in a limited neighborhood. During online fine-tuning, we incorporate more negative samples to help the student network better distinguish the target from the background where hard negative objects may exist. To this end, we crop both positive and negative samples online, as shown in Figure 5. For positive samples, we augment them through randomly flipping, shifting, increasing blur, and changing the illumination. Finally, the target patch $\mathbf{x}$ is fed into the template branch. Positive $\mathbf{z}^+$ and negative $\mathbf{z}^-$ search samples are fed into the search branch in Figure 2. For online adaptation, we jointly exploit the multi-level transfer and background-aware formulation. The online tracking loss $\mathcal{L}'_{\text{tracking}}$ is as follows:

$$\begin{aligned} \mathcal{L}'_{\text{tracking}} &= \sum_l \left(\|\mathbf{r}_l^+ - \mathbf{g}_l\|^2 + \|\mathbf{r}_l^-\|^2\right), \\ s.t. \quad \mathbf{r}_l^+ &= \mathcal{F}^{-1}\left(\hat{\mathbf{w}}_l^* \odot \hat{\varphi}_l(\mathbf{z}^+)\right), \end{aligned}$$

$$\mathbf{r}_l^- = \mathcal{F}^{-1}\left(\hat{\mathbf{w}}_l^* \odot \hat{\varphi}_l(\mathbf{z}^-)\right), \quad l \in \{\text{high, middle, low}\}, \tag{10}$$

where $+$ and $-$ on the label $\mathbf{r}$ and search patch $\mathbf{z}$ denote the positive and negative annotations, respectively. As for the fidelity loss, we do not include it in the online adaptation stage since $\mathcal{L}_{\text{fidelity}}$ is already stable after offline training and requires the heavyweight teacher model, which is unnecessary in the online tracking. The online fine-tuning is only performed on the initial frame and its loss is given as follows:

$$\mathcal{L}_{\text{online}} = \mathcal{L}'_{\text{tracking}} + \gamma \|\Theta\|^2. \tag{11}$$

C. Efficient Online Correlation Tracking

After we have the distilled lightweight backbone, we remove the additionally added 1×1 convolutional kernel, and take the output of the remaining convolutional layers to facilitate existing CF frameworks for online tracking. We select three representative methods (i.e., KCF [4], ECO [7], and STRCF [11]) as introduced in Section III.

V. Experiments

In this section, we first illustrate the implementation details and the evaluation configurations. Then we conduct an ablation study to demonstrate the effectiveness of our method. Finally, we compare with state-of-the-art trackers.

A. Experimental Details

1) Implementation Details: We use the videos for object detection from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2015) [45] dataset to offline distill the student network. During training, we use the stochastic gradient descent (SGD) solver and set the momentum and weight decay as 0.9 and 0.005, respectively. We train the network for 50 epochs with a learning rate exponentially decreased from 10^{-2} to 10^{-5}. The multi-task weighting parameter λ in Eq. (4) is set to 10^{-5}. In the online adaptation stage, we fine-tune the student network for only 8 iterations using the samples from the first frame. In each iteration, we crop 32 positive and negative samples as shown in Figure 5. We do not modify the baseline trackers and follow their implementations for feature extraction. We implement our method using MatConvNet [46] on a PC with a 4GHz CPU and an Nvidia GTX 1080TI GPU. The source code will be available at: https://github.com/594422814/CF-VGG.git

2) Benchmarks and Evaluation Metrics: We evaluate our tracker on the OTB-2015 [10], Temple-Color [47], UAV123 [48], and LaSOT-test [49] datasets, which contain 100, 128, 123 and 280 challenging videos, respectively. We report the overlap success plots on these datasets using one-pass evaluation (OPE) [10], [50] and take the area-under-curve (AUC) scores to evaluate the performance. In addition, we evaluate our tracker on the VOT-2016 [51] and VOT-2017 [52] datasets. The performance is measured by two independent metrics: accuracy (average overlap during successful tracking) and robustness (reset rate).

TABLE I

Computation Comparison Between the ECO/DeepSTRCF Trackers and Our Improved Versions on the OTB-2013 Dataset. We Use Float-Point Operations (FLOPs) of Convolution Operation to Measure the Computational Complexity, Where B Indicates Billion. In Practice, the Actual Speedup Ratio Is Much Smaller Than FLOPs

	Backbone Model	Model Size	Model FLOPs	CPU Feature Extraction	CPU FPS	GPU FPS
ECO [7]	VGG-M [41]	95 MB	1.82 B	76 ms	5	9
fECO	CF-VGG	1.5 MB	0.048 B	9 ms	22	>45
DeepSTRCF [11]	VGG-M [41]	95 MB	1.82 B	76 ms	3	5
fDeepSTRCF	CF-VGG	1.5 MB	0.048 B	9 ms	17	>33

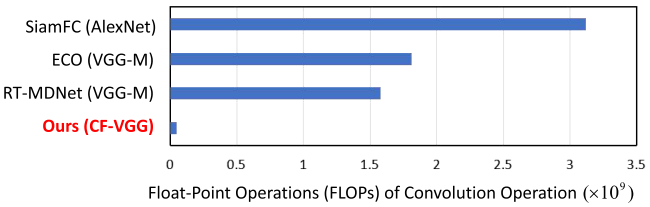


Fig. 6. Model computation complexity comparison. Our proposed lightweight CF-VGG produces much fewer FLOPs, which guarantees the CPU real-time correlation tracking.

B. Ablation Study

We evaluate the effectiveness of the components of the proposed algorithm in terms of computational efficiency, tracking accuracy, and model representation capability.

1) Efficiency: Table I compares the efficiency and model size of our CF-VGG with the original teacher network VGG-M. These two networks are integrated into the state-of-the-art CF trackers ECO and DeepSTRCF. We observe that it takes around 76 ms for the VGG-M network to extract features on the CPU, which is 8 times slower than that using our distilled CF-VGG network. The improved fECO and fDeepSTRCF trackers take 22 FPS and 17 FPS vs. their original speed 5 FPS and 3 FPS, respectively. It should be noted that the online adaptation in the first frame is an alternative choice to obtain better results. Without initial fine-tuning, our fECO and fDeepSTRCF can achieve higher speeds of 27 FPS and 20 FPS, respectively, still exhibiting competitive performance.

In addition to the comparison with CF trackers using VGG-M, we further analyze some other representative real-time trackers. Figure 6 shows the comparison results of some widely adopted backbones on FLOPs metric (only feature extraction part). The number of float-point operations (FLOPs) of the convolutional layer is calculated as follows,

$$\text{FLOPs} = (C_{\text{in}} K^2 + 1) H W C_{\text{out}}, \tag{12}$$

where C_{in} is the input feature map channel, K is the kernel width (assumed to be symmetric), $+1$ means the computation of bias operation, and H, W and C_{out} are the height, width and channel number of the output feature maps, respectively. In Table II, we exhibit the feature map sizes and feature channels of different backbone networks. The AlexNet backbone is typically used in Siamese trackers [27], [30] and the VGG-M network is widely adopted in classification based

TABLE II

COMPARISON OF THE FEATURE MAP SIZES AND FEATURE CHANNELS OF DIFERENT NETWORKS INCLUDING ALEXNET [56], VGG-M [41] AND OUR CF-VGG

	AlexNet (SiamFC [27])	VGG-M (ECO [7])	CF-VGG (fECO)
Input	$255 \times 255 \times 3$	$224 \times 224 \times 3$	$224 \times 224 \times 3$
Conv1	$123 \times 123 \times 96$	$112 \times 112 \times 96$	$112 \times 112 \times \mathbf{12}$
Pool1	$61 \times 61 \times 96$	$56 \times 56 \times 96$	$56 \times 56 \times 12$
Conv2	$57 \times 57 \times 256$	$28 \times 28 \times 256$	$28 \times 28 \times \mathbf{32}$
Pool2	$28 \times 28 \times 256$	$14 \times 14 \times 256$	$14 \times 14 \times 32$
Conv3	$26 \times 26 \times 192$	$14 \times 14 \times 512$	$14 \times 14 \times \mathbf{64}$
Conv4	$24 \times 24 \times 192$	$14 \times 14 \times 512$	$14 \times 14 \times \mathbf{64}$
Conv5	$22 \times 22 \times 128$	$14 \times 14 \times 512$	$14 \times 14 \times \mathbf{64}$

TABLE III

PERFORMANCE AND SPEED ANALYSIS ON DIFFERENT COMPRESSION RATIOS. THE BASELINE TRACKER IS ECO, AND IS EVALUATED ON THE OTB-2015 BENCHMARK [10] USING AUC METRIC. TO ACHIEVE BOTH SATISFYING CPU REAL-TIME EFFICIENCY AND PERFORMANCE, WE CHOOSE THE COMPRESSION RATE OF 64×

Compression Ratio	Baseline	1×	16×	32×	64×	96×
Model Size (MB)	95 MB	95 MB	5.8 MB	2.9 MB	1.5 MB	0.99 MB
AUC Socre (%)	69.4	69.6	69.0	68.5	68.2	66.9
CPU Speed (FPS)	5	5	8	12	22	30

TABLE IV

COMPARISON OF TRACKING ACCURACY UNDER DIFFERENT TRAINING CONFIGURATIONS. WE REPORT AUC SCORES ON THE OTB-2013 [50], OTB-2015 [10], AND TEMPLE-COLOR [47] DATASETS. THE VALUES IN BRACKETS DENOTE THE PERFORMANCE GAP COMPARED WITH THE CORRESPONDING BASELINE WITH UNCOMPRESSED DEEP MODEL

Trackers of different variations	OTB-2013	OTB-2015	TC-128
ECO (baseline)	71.0	69.4	60.3
ECOhc (hand-crafted feature)	65.6	64.6	54.7
fECO (pretrained tiny model)	66.0 (-5.0)	65.5 (-3.9)	55.4 (-4.9)
fECO (only fidelity loss)	66.1 (-4.9)	65.1 (-4.3)	55.0 (-5.3)
fECO (only tracking loss, single-layer)	65.2 (-5.8)	64.6 (-4.8)	54.6 (-5.7)
fECO (only tracking loss, multi-layer)	66.3 (-4.7)	65.9 (-3.5)	55.9 (-4.4)
fECO (fidelity + multi-layer tracking)	68.4 (-2.6)	67.9 (-1.5)	57.4 (-2.9)
fECO (offline + online fine-tune)	68.5 (-2.5)	68.2 (-1.2)	57.4 (-2.9)
DeepSTRCF (baseline)	69.2	68.5	59.9
STRCF (hand-crafted feature)	66.5	64.8	54.9
fDeepSTRCF (pretrained tiny model)	65.1 (-4.1)	65.2 (-3.3)	55.2 (-4.7)
fDeepSTRCF (only fidelity loss)	65.6 (-3.6)	65.5 (-3.0)	55.1 (-4.8)
fDeepSTRCF (only tracking loss, single-layer)	65.7 (-3.5)	65.4 (-3.1)	54.8 (-5.1)
fDeepSTRCF (only tracking loss, multi-layer)	66.9 (-2.3)	66.0 (-2.5)	55.5 (-4.4)
fDeepSTRCF (fidelity + multi-layer tracking)	69.4 (+0.2)	67.8 (-0.7)	56.9 (-3.0)
fDeepSTRCF (offline + online fine-tune)	70.3 (+1.1)	68.6 (+0.1)	57.3 (-2.6)

trackers [53]–[55] and CF trackers [7], [11]. After computing the FLOPs of different backbone networks via Eq. (12), we can observe that our tiny model is extremely efficient than modern off-the-shelf models. The FLOPs of the feature extractor in SiamFC and ECO are 3.12×10^9 and 1.82×10^9 while ours is only 4.79×10^7, as shown in Figure 6.

The Siamese trackers [27], [30], [57], [58] adopt AlexNet-like [56] fully-convolutional networks to predict target location in an end-to-end manner. Their tracking speed can be significantly accelerated to over 80 FPS by a powerful GPU because the fully-convolutional structure adequately exploits the GPU device. However, on a single CPU, the Siamese trackers are unlikely to achieve real-time performance [34], whereas our improved CF trackers can. The recent real-time MDNet tracker [59] modifies the first three convolutional layers of VGG-M and uses ROI Align for efficient binary classification. However, its backbone network still produces high FLOPs and the further online fine-tune prevents its CPU real-time performance. For most deep CF trackers (e.g., HCF [6], ECO [7] and STRCF [11]), only the deep feature extraction process benefits from GPU and the tracking part just uses CPU even without optimization. Besides, ECO and STRCF methods use a time-consuming alternating direction method of multipliers (ADMM) or Conjugate Gradient (CG) for online algorithm optimization. Thus, existing speed comparison that does not distinguish CPU and GPU environments is not very fair. With only CPU, the efficient Siamese tracker (e.g., SiamFC, about 86 FPS on a GPU) cannot achieve real-time speed [34] but ours can make it, which already proves the efficiency of CF trackers using our tiny model.

2) Compression Ratio: In this work, we compress the off-the-shelf model by an extremely high ratio of about 63×. In Table III, we evaluate the performance, model size and CPU speed under different network compression ratios. The 1× compression ratio means that the network is not pruned, but still fine-tuned by the fidelity loss and correlation tracking loss. It slightly outperforms the teacher model, which shows that our joint training scheme is effective and the tracking loss slightly fine-tunes the uncompressed model. To achieve CPU real-time speed, we choose the compression ratio of 64×. Except for the better efficiency, by adopting our lightweight model, the required storage room is also greatly saved (our 1.5 MB vs. original 95 MB).

3) Tracking Accuracy: In Table IV, we show the performance evaluation results using different configurations to distill the student network. To obtain a tiny model, an optional choice is directly training a tiny CF-VGG from scratch using object classification loss (i.e., cross-entropy loss) following original VGG-M (denoted as "pretrained tiny model" in Table IV), but its performance is unsatisfied since it may not suit the tracking task. In contrast, we propose to jointly compress and transfer a teacher network. When using only the fidelity loss (shown as "only fidelity loss" in Table IV), the tracking accuracy decreases by 4∼5% for ECO. Meanwhile, using only tracking loss (denoted as "only tracking loss" in Table IV) decreases the accuracy by 3∼4% as well. However, equipped with both the fidelity and tracking losses (denoted as "fidelity + multi-layer tracking" in Table IV), we significantly improve the performance, which means the high-level semantic features can complement the multi-level appearance features trained via tracking loss. When integrated into DeepSTRCF, we find that the improved fDeepSTRCF tracker achieves higher accuracy on both the OTB-2013 and OTB-2015 datasets. Our performance slightly decreases on the Temple-Color dataset.

The aforementioned versions do not consider the model fine-tuning using the initial background context. Finally, based on the "fidelity + multi-layer tracking" version, we further improve it by adding online adaptation (i.e., "offline + online fine-tune" in Table IV), the trackers show slightly better results and the performance gap is only about 1∼2% compared to the baselines with uncompressed deep features. In addition, our improved fECO and fDeepSTRCF trackers achieve much higher performance than ECOhc and STRCF, which both use

TABLE V

FEATURE REPRESENTATION CAPABILITY COMPARISON BETWEEN VGG-M AND OUR COMPRESSED MODEL. WE PRESENT AUC SCORES ON THE OTB-2015 [10] DATASET. OUR FKCF ACHIEVES COMPARABLE PERFORMANCE ON EACH SINGLE FEATURE LAYER WITH ITS TEACHER

	Backbone Model	Model Size	Low-level Conv 1	Middle-level Conv 2	High-level Conv 5	CPU FPS
KCF	VGG-M [41]	95 MB	48.0	50.6	49.2	6
fKCF	CF-VGG	1.5 MB	46.8 (-1.2)	51.0 (+0.4)	47.1 (-2.1)	42

hand-crafted features. Finally, as shown in Table IV, it is worth mentioning that the offline pretrained CF-VGG model already works well even without online adaptation.

4) Model Representation Capability: The state-of-the-art CF trackers (i.e., ECO and DeepSTRCF) employ spatial regularization to reduce boundary effects in learning correlation filters. This may leave the concern that how likely the compressed CF-VGG model can maintain the presentation capability of deep models. To demonstrate the effectiveness of CF-VGG, we use the baseline KCF method [4] to evaluate the performance on each single feature level. Table V shows that KCF with CF-VGG exhibits comparable performance with the original teacher network. This indicates that the distilled student network almost maintains the same feature representation capability even though its model size is 63 times smaller than its teacher network.

C. Comparison With State-of-the-Art Methods

1) OTB-2015 Dataset: OTB-2015 [10] is a popular tracking benchmark with 100 challenging videos. On this dataset, we compare our fECO and fDeepSTRCF with 20 state-of-the-art trackers, which are mainly categorized as real-time trackers and non-realtime trackers.

- Real-time Trackers: For comprehensive comparison, we collect recent high-performance real-time trackers including TRACA [37] (100 FPS), SiamRPN [32] (160 FPS), BACF [60] (35 FPS), CFNet [28] (65 FPS), CSR-DCF [16] (15 FPS), ACFN [61] (15 FPS), SiamFC [27] (86 FPS), Staple [23] (70 FPS), SCT4 [62] (50 FPS), and KCF [4] (270 FPS). It should be noted that some of these trackers require GPU to achieve high speed (e.g., TRACA, SiamRPN, CFNet, ACFN, and SiamFC). In contrast, our methods are free of such requirement.
- Non-realtime Trackers: We compare with high accuracy trackers including VITAL [54] (1.5 FPS), DSLT [63] (5 FPS), CREST [64] (3 FPS), MCPF [65] (2 FPS), ADNet [66] (1 FPS), C-COT [17] (0.3 FPS), MDNet [53] (1 FPS), SRDCFdecon [67] (3 FPS), DeepSRDCF [42] (<1 FPS), and HCF [6] (12 FPS). Among these trackers, only C-COT and SRDCFdecon are tested on CPU and all the other trackers rely on a high-end GPU. Although these trackers achieve state-of-the-art performance on the benchmarks, their computational load limits the practical usage. In the following experiments, we will show that our CPU real-time methods still outperform most of them.

On the OTB-2015, our fECO and fDeepSTRCF exhibit the AUC scores of 68.2% and 68.6%, respectively. Figure 7 shows

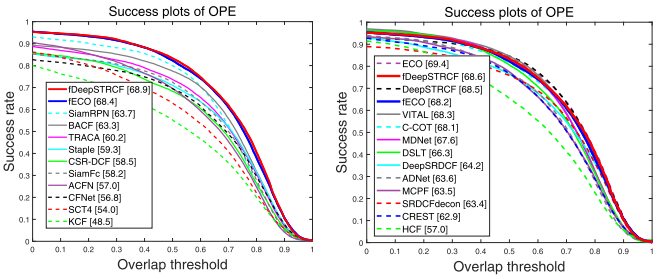


Fig. 7. Success plots of real-time trackers (left) and non-realtime trackers (right) on the OTB-2015 [10] dataset. Our trackers surpass other real-time methods and even outperform most non-realtime deep trackers.

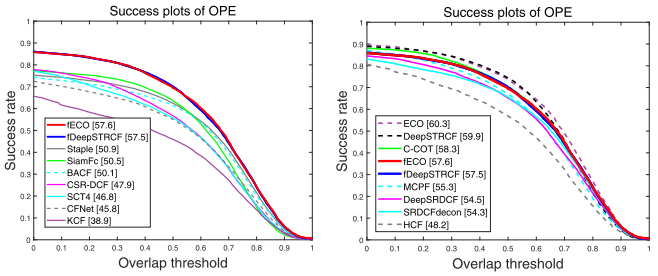


Fig. 8. Success plots of real-time trackers (left) and non-realtime trackers (right) on the Temple-Color [47] dataset. Our trackers show outstanding performance among real-time trackers and comparably favorable results among non-realtime trackers.

that our methods outperform the recent real-time trackers and perform favorably against non-realtime deep trackers. The TRACA tracker [37] uses an encoder network to reduce the feature channel and achieves high speed on GPU. In contrast, our CF-VGG not only reduces feature dimension but also greatly accelerates the feature extraction time, which brings in real-time speed on the CPU and better performance (about 8% higher in AUC). The recent SiamRPN [32] improves the SiamFC tracker [27] and achieves impressive performance. However, it needs GPU to achieve high speed and our CPU real-time methods still outperform it by about 5% in AUC score. The deep feature representation of CF-VGG enables our trackers to surpass traditional CF trackers using empirical features (e.g., BACF [60], Staple [23], and CSR-DCF [16]). Furthermore, our methods even outperform many recent deep trackers that run at only 1 FPS on GPU (e.g., VITAL [54] and MDNet [53]).

2) Temple-Color Dataset: We further evaluate our trackers on the Temple-Color benchmark [47] with 128 color videos. On the Temple-Color, our fECO and fDeepSTRCF yield the AUC scores of 57.4% and 57.3%, respectively. From the left figure in Figure 8, we can observe that our trackers perform better than state-of-the-art real-time trackers (e.g., BACF [60], Staple [23] and SiamFC [27]). Compared with the non-realtime deep trackers including C-COT [17] and MCPF [65], the improved fECO and fDeepSTRCF trackers achieve comparable performance.

3) VOT-2016 and VOT-2017 Datasets: We further compare our trackers with state-of-the-art methods on the VOT-2016 [51] and VOT-2017 [52] benchmarks. On the VOT

TABLE VI

THE EXPECTED AVERAGE OVERLAP (EAO) OF STATE-OF-THE-ART METHODS ON THE VOT-2016 [51] (LEFT) AND VOT-2017 [52] (RIGHT) DATASETS. THE COMPARATIVE METHODS INCLUDE THE TOP PERFORMERS ON BOTH DATASETS, OUR BASELINE METHODS (ECO [7] AND DEEPSTRCF [11]) AND THE RECENTLY PROPOSED TRACKERS

	Trackers	EAO
Non-realtime	ECO [7]	0.374
	DSLT [63]	0.332
	VITAL [54]	0.323
	FlowTrack [68]	0.334
	DeepSTRCF [11]	0.313
	C-COT [17]	0.331
	MDNet [53]	0.227
Real-time Trackers	SiamRPN [32]	0.344
	SA-Siam [30]	0.291
	StructSiam [57]	0.264
	MemTrack [69]	0.273
	ECOhc [7]	0.238
	STRCF [11]	0.279
	BACF [60]	0.233
	Staple [23]	0.295
	SiamFC [27]	0.277
	fDeepSTRCF	0.308
	fECO	0.339

	Trackers	EAO
Non-realtime	LSART [70]	0.323
	CFCF [71]	0.286
	ECO [7]	0.280
	C-COT [17]	0.267
	MCPF [65]	0.248
	DeepSTRCF [11]	0.227
	DLST [52]	0.233
Real-time Trackers	SiamRPN [32]	0.243
	SiamDCF [52]	0.249
	SA-Siam [30]	0.236
	CSRDCF++ [16]	0.229
	ECOhc [7]	0.238
	STRCF [11]	0.162
	UCT [52]	0.206
	Staple [23]	0.169
	SiamFC [27]	0.188
	fDeepSTRCF	0.214
	fECO	0.255

benchmark, a tracker will be re-initialized when tracking failure occurs. The expected average overlap (EAO) is the evaluation metric which considers both the tracking accuracy (overlap with the ground truth box) and robustness (failure times) [72]. As shown in Table VI, our methods outperform ECOhc and STRCF. This affirms that CF-VGG performs favorably against empirical features. In addition, our trackers achieve comparable or even better results than the VOT-2016 top performer C-COT [17], whose running speed is only 0.3 FPS on a CPU. Compared with other state-of-the-art and recently proposed trackers (e.g., SA-Siam [30], VITAL [54], DSLT [63], SiamRPN [32], FlowTrack [68]), our methods overall show competitive performance.

4) UAV123 Dataset: Our lightweight model is suitable for object tracking in aerial videos since most UAV platforms do not have powerful GPU devices. We test our fECO and fDeepSTRCF on the UAV123 dataset [48], which consists of 123 low-attitude aerial videos collected by a UAV. As shown in Figure 9, our fECO and fDeepSTRCF are on par with their original deep trackers with uncompressed VGG models. For example, our fDeepSTRCF sacrifices less than 1% DP and AUC scores but operates about 5× faster on a CPU.

5) LaSOT Dataset: Finally, we evaluate our trackers with state-of-the-art methods on the recently released LaSOT benchmark [49]. LaSOT is a large-scale dataset containing 1200 videos. The average video length in LaSOT is about 2500 frames, which is more challenging than previous short-term benchmarks such as OTB and VOT. We evaluate our methods on the test set of LaSOT with 280 videos. On this benchmark, our fast trackers still exhibit similar performance with their original versions. As shown in Figure 10, ECO and DeepSTRCF slightly outperform our fECO and fDeepSTRCF by 1.3% and 0.3%, respectively, which affirms that our light-weight student network maintains the representational power of the teacher model.

6) Attribute Evaluation: All the 100 videos in OTB-2015 [10] are annotated with 11 different attributes,

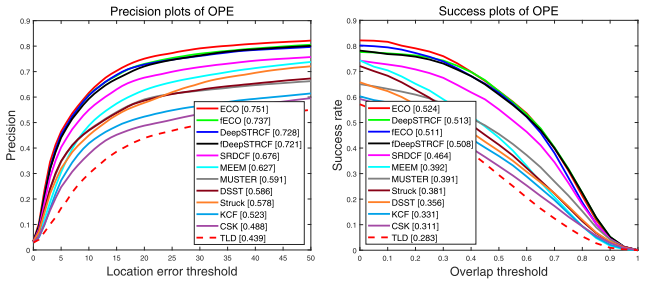


Fig. 9. Precision and success plots on the UAV123 dataset [48]. In the legend, the DP at a threshold of 20 pixels and AUC scores are reported.

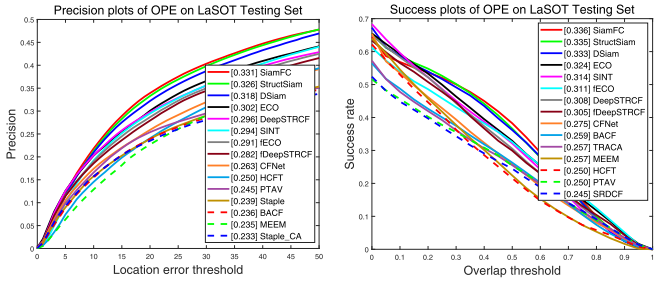


Fig. 10. Precision and success plots on the LaSOT-test dataset [49]. In the legend, the DP at a threshold of 20 pixels and AUC scores are reported.

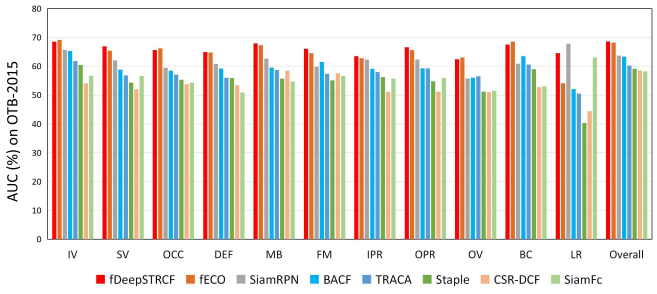


Fig. 11. Attribute-based evaluation on the OTB-2015 benchmark [10]. The evaluation metric is the area-under-curve (AUC) score of the success plot. We also put the overall performance here (the last one) for comparison convenience facing a single challenge and their combination. Only the top 6 real-time trackers are displayed for clarity. Our fDeepSTRCF and fECO algorithms perform favorably against state-of-the-art real-time trackers in various challenging scenes.

namely: background clutter (BC), deformation (DEF), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), illumination variation (IV), motion blur (MB), in-plane rotation (IPR), out of view (OV), fast motion (FM) and low resolution (LR).

In Table VII and Figure 11, we show the comparison results of 10 real-time trackers (these trackers are from Section V-C) when facing the above challenging factors. The results show that our fECO and fDeepSTRCF trackers outperform other competitors in almost all the challenging scenes.

In Table VIII, we further compare our methods with their teachers (i.e., ECO and DeepSTRCF) on attributed videos. From the results, we can observe that our compressed model is comparable with the teacher model in most attributes, but performs not good enough in fast motion (FM), motion blur

TABLE VII

ATTRIBUTE-BASED EVALUATION ON THE OTB-2015 BENCHMARK [10]. THE EVALUATION METRIC IS THE AREA-UNDER-CURVE (AUC) SCORE OF THE SUCCESS PLOT. THE FIRST AND SECOND HIGHEST VALUES ARE HIGHLIGHTED BY BOLD AND UNDERLINE

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR	Overall
TRACA [37]	61.8	56.8	57.1	56.0	58.7	57.4	58.0	59.3	56.5	60.6	50.5	60.2
SiamRPN [32]	65.7	62.0	59.4	60.8	62.6	59.8	62.3	62.3	55.8	60.9	**67.8**	63.7
BACF [60]	65.3	58.8	58.4	59.2	59.5	61.5	59.1	59.3	56.0	63.5	52.0	63.3
CFNet [28]	54.2	54.7	51.6	47.7	54.5	55.0	56.9	54.4	41.9	55.6	63.5	56.8
CSR-DCF [16]	54.0	52.0	53.8	53.4	58.4	57.5	51.1	51.1	51.0	52.7	44.4	58.5
ACFN [61]	56.5	56.3	54.5	53.6	56.4	57.0	54.5	54.6	51.4	54.8	52.0	57.0
SiamFC [27]	56.7	56.6	54.3	50.9	54.6	56.6	55.7	55.9	51.4	53.0	63.0	58.2
Staple [23]	60.4	54.3	55.3	55.9	55.7	55.1	56.2	54.8	51.2	59.0	40.3	59.1
SCT4 [62]	52.6	44.2	50.5	51.2	53.0	54.1	52.8	51.8	43.7	55.6	29.0	53.8
KCF [4]	48.7	39.7	44.9	44.5	46.8	46.6	47.6	45.9	39.7	50.4	28.8	48.3
fECO	**69.1**	65.4	**66.2**	64.8	67.3	64.5	62.8	65.6	**63.0**	**68.5**	54.1	68.2
fDeepSTRCF	68.5	**66.9**	65.6	**64.9**	**67.9**	**66.1**	**63.5**	**66.6**	62.4	67.5	64.5	**68.6**

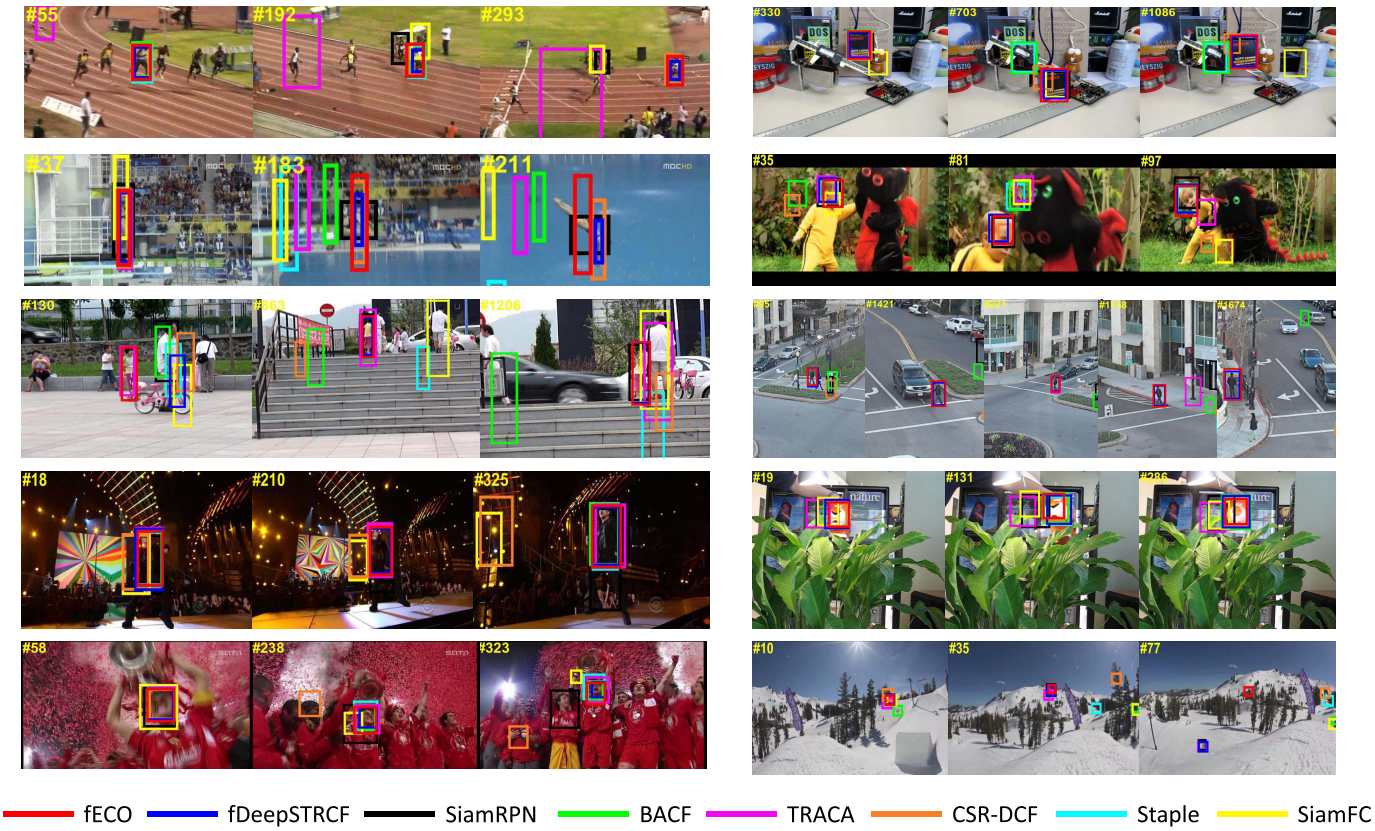


Fig. 12. Qualitative evaluation of our trackers (e.g., fECO and fDeepSTRCF) and six other state-of-the-art real-time trackers including SiamRPN [32], BACF [60], TRACA [37], CSR-DCF [16], Staple [23] and SiamFC [27] on 10 challenging sequences (from left to right and top to down: *Bolt2, Box, Diving, DragonBaby, Girl2, Human3, Singer2, Tiger1, Soccer* and *Skiing*, respectively). Our fECO and fDeepSTRCF trackers perform favorably against the state-of-the-art methods.

(MB), out of view (OV) and low resolution (LR), which indicates the representation capability of our student network still has improvement room. It should be noted that the model size of our network is only 1/63 of its teacher, so the slight performance degradation is bearable since our trackers achieve superiorly balanced high performance and CPU real-time efficiency.

7) Qualitative Evaluation: Figure 12 shows some comparison results of our trackers (fECO and fDeepSTRCF) and other six state-of-the-art real-time trackers including SiamRPN [32], BACF [60], TRACA [37], CSR-DCF [16], Staple [23] and SiamFC [27] on ten challenging sequences. From the results in Figure 3, we can see that our fECO and fDeepSTRCF trackers perform well on occlusion (e.g., *Box, Girl2, Human3* and *Soccer*) and background clutter (e.g., *Tiger1* and *Soccer*). Compared with the recent real-time deep trackers (SiamRPN [32] and TRACA [37]), our methods perform favorably against them while exhibiting the CPU real-time speed.

D. Failure Cases

Finally, we show some failure cases of our method in Figure 13. In the video *Freeman4*, the target with low resolution undergoes frequent occlusions in a short span of time, while the *Ironman* in the second video occurs a drastic appearance change. In these cases, our compressed model is not powerful enough compared to the teacher network. In our

TABLE VIII

ATTRIBUTE-BASED EVALUATION BETWEEN OUR METHODS AND THEIR CORRESPONDING BASELINES (ECO [7] AND DEEPSTRCF [11]) WITH UNCOMPRESSED NETWORKS. THE AUC SCORE IS REPORTED ON THE OTB-2015 DATASET [10]

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR	Overall
ECO [7]	71.2	68.2	68.0	63.4	70.4	68.1	65.4	67.5	67.1	71.2	58.1	69.4
fECO	69.1	65.4	66.2	64.8	67.3	64.5	62.8	65.6	63.0	68.5	54.1	68.2
Δ	-2.1	-2.8	-1.8	+1.4	-3.1	-3.6	-2.6	-1.9	-4.1	-2.7	-4.0	-1.2
DeepSTRCF [11]	67.5	66.8	66.2	64.1	68.3	66.5	62.9	66.6	64.8	64.6	63.7	68.5
fDeepSTRCF	68.5	66.9	65.6	64.9	67.9	66.1	63.5	66.6	62.4	67.5	64.5	68.6
Δ	+1.0	+0.1	-0.6	+0.8	-0.4	-0.4	+0.6	0	-2.4	+2.9	+0.8	+0.1

Fig. 13. Failure cases of the proposed method. The videos are *Ironman* and *Freeman4* from OTB-2015 [10]. Our compressed model struggles when an occlusion or a drastic appearance change occurs.

future work, we aim to include more training data and adopt a better network structure to further enhance the representation capability of the student model.

VI. CONCLUSION

In this paper, we propose to learn a lightweight backbone network for real-time correlation tracking. By simultaneously compressing and transferring the teacher network pre-trained on object recognition, we obtain a highly compressed lightweight model (63× smaller) as the feature backbone. Extensive experiments demonstrate that our training scheme and strategies are effective and efficient. Even though being extremely lightweight, the proposed distilled backbone network is sufficiently powerful and almost maintains the same feature representation capability as the teacher network. Leveraging our lightweight model for deep correlation tracking, the recent top CF trackers consume much less memory storage and show superiorly balanced high performance and CPU real-time efficiency.

REFERENCES

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.

[2] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[3] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550.

[4] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[5] K. Zhang, L. Zhang, M.-H. Yang, and D. Zhang, "Fast visual tracking via dense spatio-temporal context learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2013, pp. 127–141.

[6] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3074–3082.

[7] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6638–6646.

[8] G. Bhat, J. Johnander, M. Danelljan, F. S. Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 483–498.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[10] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[11] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning spatial-temporal regularized correlation filters for visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4904–4913.

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: http://arxiv.org/abs/1503.02531

[13] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*. [Online]. Available: http://arxiv.org/abs/1412.6550

[14] F. Liu, C. Gong, X. Huang, T. Zhou, J. Yang, and D. Tao, "Robust visual tracking revisited: From correlation filter to template matching," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2777–2790, Jun. 2018.

[15] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4310–4318.

[16] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6309–6318.

[17] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 472–488.

[18] C. Sun, D. Wang, H. Lu, and M.-H. Yang, "Correlation tracking via joint discrimination and reliability learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 489–497.

[19] M. Tang, B. Yu, F. Zhang, and J. Wang, "High-speed tracking with multi-kernel correlation filters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4874–4883.

[20] T. Zhang, S. Liu, C. Xu, B. Liu, and M.-H. Yang, "Correlation particle filter for visual tracking," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2676–2687, Jun. 2018.

[21] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5388–5396.

[22] N. Wang, W. Zhou, and H. Li, "Reliable re-detection for long-term tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 730–743, Mar. 2019.

[23] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1401–1409.

[24] N. Wang, W. Zhou, Q. Tian, R. Hong, M. Wang, and H. Li, "Multi-cue correlation filters for robust visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4844–4853.

[25] K. Zhang, J. Fan, Q. Liu, J. Yang, and W. Lian, "Parallel attentive correlation tracking," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 479–491, Jan. 2019.

[26] Y. Qi *et al.*, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4303–4311.

[27] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 850–865.

[28] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-End representation learning for correlation filter based tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2805–2813.

[29] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: Residual attentional siamese network for high performance online visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4854–4863.

[30] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4834–4843.

[31] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, "Unsupervised deep tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1308–1317.

[32] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8971–8980.

[33] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018. pp. 101–117.

[34] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 105–115.

[35] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–5.

[36] M. Danelljan, F. S. Khan, M. Felsberg, and J. V. D. Weijer, "Adaptive color attributes for real-time visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1090–1097.

[37] J. Choi *et al.*, "Context-aware deep feature compression for high-speed visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 479–488.

[38] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Peter Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: http://arxiv.org/abs/1608.08710

[39] J. Ba and R. Caruana, "Do deep nets really need to be deep," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 2654–2662.

[40] G. Zhu, J. Wang, P. Wang, Y. Wu, and H. Lu, "Feature distilled tracking," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 440–452, Feb. 2019.

[41] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2014.

[42] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 58–66.

[43] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, "DCFNet: Discriminant correlation filters network for visual tracking," 2017, *arXiv:1704.04057*. [Online]. Available: http://arxiv.org/abs/1704.04057

[44] M. Mueller, N. Smith, and B. Ghanem, "Context-aware correlation filter tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1396–1404.

[45] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[46] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia MM*, 2015, pp. 689–692.

[47] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: Algorithms and benchmark," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5630–5644, Dec. 2015.

[48] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 445–461.

[49] H. Fan *et al.*, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5374–5383.

[50] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2411–2418.

[51] M. Kristan *et al.*, "The visual object tracking vot2016 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshops (ECCV Workshop)*, Dec. 2016, pp. 1–23.

[52] M. Kristan *et al.*, "The visual object tracking VOT2017 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1949–1972.

[53] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4293–4302.

[54] Y. Song *et al.*, "VITAL: VIsual tracking via adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8990–8999.

[55] S. Pu, Y. Song, C. Ma, H. Zhang, and M.-H. Yang, "Deep attentive tracking via reciprocative learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 1931–1941.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2012, pp. 1097–1105.

[57] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, "Structured siamese network for real-time visual tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 351–366.

[58] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1763–1771.

[59] I. Jung, J. Son, M. Baek, and B. Han, "Real-time mdnet," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 83–98.

[60] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1135–1143.

[61] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, and J. Y. Choi, "Attentional correlation filter network for adaptive visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4807–4816.

[62] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, "Visual tracking using attention-modulated disintegration and integration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4321–4330.

[63] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang, "Deep regression tracking with shrinkage loss," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 353–369.

[64] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. H. Lau, and M.-H. Yang, "CREST: Convolutional residual learning for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2555–2564.

[65] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4335–4343.

[66] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2711–2720.

[67] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1430–1438.

[68] Z. Zhu, W. Wu, W. Zou, and J. Yan, "End-to-End flow correlation tracking with spatial-temporal attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 548–557.

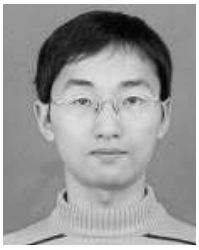
[69] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 152–167.

[70] C. Sun, D. Wang, H. Lu, and M.-H. Yang, "Learning spatial-aware regressions for visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8962–8970.

[71] E. Gundogdu and A. A. Alatan, "Good features to correlate for visual tracking," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2526–2540, May 2018.

[72] M. Kristan *et al.*, "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2137–2155, Nov. 2016.

Ning Wang received the B.E. degree in communication engineering from Tianjin University (TJU), in 2016. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineer and Information Science, University of Science and Technology of China (USTC). His research interest is computer vision. His current research work is focused on video object tracking.



Wengang Zhou (Member, IEEE) received the B.E. degree in electronic information engineering from Wuhan University, China, in 2006, and the Ph.D. degree in electronic engineering and information science from the University of Science and Technology of China (USTC), China, in 2011. From September 2011 to September 2013, he worked as a Postdoctoral Researcher with the Computer Science Department, The University of Texas at San Antonio. He is currently a Professor with the EEIS Department, USTC. His research interests include multimedia information retrieval and computer vision.

Yibing Song received the bachelor's degree from the University of Science and Technology of China and the Ph.D. degree from the City University of Hong Kong in 2018. He is currently a Researcher with the Tencent AI Lab. His research interests reside in object tracking and face analysis.

Chao Ma (Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University in 2016. He was a Senior Research Associate with the Australian Centre of Robotic Vision, The University of Adelaide, from 2016 to 2018. He has been an Assistant Professor with Shanghai Jiao Tong University since 2019. His research interests include computer vision and machine learning. He was sponsored by the China Scholarship Council as a visiting Ph.D. Student with the University of California at Merced from 2013 to 2015.

Houqiang Li (Senior Member, IEEE) received the B.S., M.Eng., and Ph.D. degrees in electronic engineering from the University of Science and Technology of China, Hefei, China, in 1992, 1997, and 2000, respectively.

He is currently a Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include multimedia search, image/video analysis, video coding, and communication. He has authored or coauthored more than 200 articles in journals and conferences. He is the winner of National Science Funds (NSFC) for Distinguished Young Scientists, the Distinguished Professor of Changjiang Scholars Program of China, and the Leading Scientist of Ten Thousand Talent Program of China. He was a recipient of the National Technological Invention Award of China (second class) in 2019 and the National Natural Science Award of China (second class) in 2015. He was also a recipient of the Best Paper Award for VCIP 2012, the Best Paper Award for ICIMCS 2012, and the Best Paper Award for ACM MUM in 2011. He has served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY from 2010 to 2013. He has served as the TPC Co-Chair of VCIP 2010, and he will serve as the General Co-Chair of ICME 2021.