A Collaborative Visual Tracking Architecture for Correlation Filter and Convolutional Neural Network Learning

Wei Tian, Niels Ole Salscheider, Yunxiao Shan, Long Chen, and Martin Lauer

***Abstract*— Visual object tracking has achieved remarkable progress in recent years and has been broadly applied in intelligent transportation systems such as autonomous vehicles and drones to monitor and analyze the behavior of specific targets. One typical tracking approach is the discriminative tracker, which branches into two main categories: the correlation filter (CF) and the convolutional neural network (CNN). However, most of the current researches consider both categories as two separate techniques and only rely on one of them. Thus, a dense cooperation between the CF and the CNN still remains less discovered and the question of how to effectively join both techniques to further boost the tracking performance is still open. To address this issue, in this paper, we propose a collaborative architecture which incorporates models constructed with both techniques and dynamically aggregates their response maps for target inference. By an alternating optimization, both models are learned on each other's errors to persistently improve the classification power of the whole tracker. For further efficiency, we present a faster solver for our utilized CF and an analytical solution for dynamic model weighting. Through experiments on standard benchmarks, we reveal the influence of key factors on the joint learning architecture and show that it outperforms the state-of-the-art approaches.**

***Index Terms*— Collaborative visual tracking, correlation filter (CF), convolutional neural network (CNN), joint learning.**

I. Introduction

IN RECENT years, visual perception techniques become unprecedentedly popular among automobile industries thanks to breakthroughs from the computer vision, pattern recognition and machine learning domain. As one of its essential branches, the visual object tracking approach has been broadly embedded into advanced driver assistance systems (ADAS), autonomous driving vehicles, drones, and surveillance devices on infrastructures for monitoring and analyzing the behavior of specific targets. Normally, in the visual tracking approach, given a query image, the tracker is initialized by extracting visual features from the predefined target region (usually as a bounding box provided by an external detector) to construct the appearance model. Thereupon, it is applied on following video frames to seek the most similar image sample as the target and updated with respect to the new object appearance.

Recently, the discriminative tracking approach has been drawing increased attention from the research community. By introducing powerful machine learning techniques, the tracking procedure can be reformulated as a classification process, significantly improving the robustness against appearance variations. Currently, depending on which kind of classifier is employed, the discriminative tracking approach branches into two main categories: the correlation filter (CF) and the convolutional neural network (CNN).

The correlation filter takes over the idea of the structured support vector machine (SVM) [1]. On the one hand, the filter is trained in a regression fashion with soft labels, which are presented by a 2D Gaussian map with values in a range from zero to one. On the other hand, circularly shifted samples are extracted for both training and classification. Thus, the calculation in the Fourier domain becomes more efficient. Such filter structure has been adopted in lots of tracking works [2]–[4] and is demonstrated with high robustness against image noise like background clutter and motion blur.

Along with the resurgence of deep learning in recent years, the convolutional neural network has become a promising approach. The CNN is able to integrate both feature extraction and model construction in one training procedure. Hence, various CNNs have been well studied in learning sophisticated models and robust feature representations, especially against rotation and deformation of tracked targets [5], [6].

A recent study has shown that CF and CNN based trackers can yield comparable performance [7]. However, as the difference in modeling procedure is significant, most prior works still consider both approaches as two separate techniques and only rely on one of them. Furthermore, the combination of both approaches explored in current works is relative sparse, e.g., by introducing deep features into CFs [8] or training CNN layers with circularly shifted samples [9]. Thus, a dense cooperation between CF and CNN still remains less discovered and the question of how to effectively join both techniques to further boost the tracking performance is still open.

Manuscript received July 17, 2018; revised November 9, 2018, February 15, 2019, and April 16, 2019; accepted July 12, 2019. Date of publication July 25, 2019; date of current version July 29, 2020. The work of Y. Shan was supported by the Research and Development Plan in Key Areas of Guangdong Province under Grant 2018B010108004 and 2019B090919003. The Associate Editor for this paper was C. Guo. *(Corresponding author: Yunxiao Shan.)*

W. Tian, N. O. Salscheider, and M. Lauer are with the Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: wei.tian@kit.edu; ole.salscheider@kit.edu; martin.lauer@kit.edu).

Y. Shan and L. Chen are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: shanyx@mail.sysu.edu.cn; chenl46@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TITS.2019.2928963

As acknowledged, both the CF and CNN are based on the convolution operation between the filter coefficient and the image. By utilizing the L_2-norm for the error term, mathematically they share a similar loss function form (shown later in the paper). However, the difference between CF and CNN lies mainly in the optimization procedure. The CF reformulates the learning procedure in the frequency domain and analytically solves the filter coefficients. Conversely, the CNN directly optimizes the filter coefficients by the Back Propagating approach with the stochastic gradient descent. To accommodate these issues, in this paper, we propose a collaborative architecture incorporating models constructed with both CF and CNN, which are learned on the same image region. Leveraging the model convexity, they are jointly trained in a way of alternating optimization. The target inference is obtained by dynamically aggregating their response maps. For processing efficiency, a faster solver is utilized for the CF and an analytical solution is provided for the dynamic model weighting. Additionally, the proposed approach is validated on standard benchmarks. To sum up, our contributions are threefolds:

- We propose a framework to join both the CF and CNN tracker based on two facts. At first, both kinds of trackers employ the convolution operation to evaluate the input image with filter coefficients. Secondly, by utilizing L_2-norm, both kinds of trackers mathematically share a common loss function form. These facts make it possible to integrate both tackers into a general multi-filter learning architecture.
- We propose an alternate optimization scheme to jointly learn both CF and CNN trackers. In this scheme, both trackers are persistently learned from the error of each other. Their responses are aggregated for the final inference and the corresponding weights are efficiently optimized in the scheme by an analytical solution.
- We reveal the influence of key factors on our joint tracking architecture. Through experiments on standard benchmarks, we demonstrate that our approach performs superior over state-of-the-art trackers. This work is also promising to be generalized for various kinds of CFs and CNNs.

II. Related Work

Visual object tracking has enjoyed a rapid development in the last decade and relevant surveys can be referred to literatures such as [7], [10].

In early works, trackers mostly rely on generative models with predefined features and search the most similar candidate by matching operation w.r.t. specific probabilistic models or distance metrics. Thus, the focus of their work is in developing a reliable visual representation of the target. For instance, color histograms of small image regions are extracted to match tracked object by [11]. In work [12], intensity-based image templates are calculated by an incremental algorithm of principal component analysis (PCA) to represent targets. However, these methods exhibit insufficient effectiveness, especially when the target undergoes large appearance variation such as motion blur, rotation, etc.

In order to tackle these challenges, knowledge from the machine learning domain has been introduced in the object tracking community to boost the discriminative power of tracker models, consequently yielding the discriminative tracking approach. Despite the fact that conventional classification methods like boosting [13] and random forests [14], [15] are adopted in some prior works, due to issues of processing efficiency and robustness of feature representation, current researches mainly focus on two approaches: the correlation filter and the convolutional neural network.

The correlation filter stems from the structured SVM. As the label regression is performed for circularly shifted versions of the input image patch, a large amount of sampling work can be spared. Due to its efficiency, CF based trackers receive considerable attention and plenty of strategies have been proposed to boost its performance. For example, kernel tricks are suggested by [16], yielding a favorable precision gain. Multi-dimensional features like color attributes [17] are adopted to enhance appearance models. Scale estimation [18] is also deployed to cope with size variations. In other extensions, manipulations are oriented on training samples, such as multi-memory domains [19], pixel priors [20] and background awareness [21]. To obtain a robust visual representation, deep features are introduced into tracker models [8], [22]. However, since the CNN is only exploited as a feature extractor in these works, the cooperation between CF and CNN still remains on the feature level.

Due to the recent success of deep learning in visual tasks like object classification and scene recognition, researchers are also motivated to adopt CNNs to implement trackers, which are diverse in both feature extractions and model structures. Representatively, the work [23] combines responses from various convolutional layers to promote the robustness against large scale variations. The work [24] proceeds the classification layer with pre-trained common layers to take advantage of knowledge learned from large video domains. For object identification between two input streams, the Siamese network is adopted in cooperation with localization by regression [25], fully-convolutional architecture [26], learned matching function [27], etc. The pair-wise identification is extended in [28] by a verification scheme to measure multi-stream similarities. Nevertheless, the above discussed approaches are solely CNN based trackers. In recent studies, works such as [29], [30] attempt to learn more discriminative features from CNN to apply on CF trackers. Aside from that, the study [9] trains the CNN with circularly shifted samples, proving an improved efficiency. However, this procedure is mainly focused on the initialization step and the topic of a more effective combination of both CF and CNN models is not covered.

Compared with prior works, we propose a collaborative architecture incorporating both CF and CNN models. By emphasizing the convexity, they can be jointly learned in an alternating optimization fashion. And each model is trained on each other's errors to persistently boost the classification power of the whole tracker. As the target inference is obtained by a dynamic aggregation of their individual responses,

the approach exhibits a superior performance against large appearance variations.

III. Proposed Method

In this section, we firstly derive our collaborative architecture from the reformulation of multi-filter learning. In the meanwhile, we describe the entire learning procedure for this architecture. Furthermore, we introduce the structure of each individual classifier as well as a faster solver for the utilized CF tracker. Afterwards, we give a discussion about the online tracking and model updating.

A. Collaborative Learning Architecture

The correlation filter is dedicated to locate the target based on a dense response map, which is yielded by correlating the learned coefficient vector $\mathbf{w}$ with all shifted versions of an image patch $\mathbf{x}$. Here we revisit its linear form and interpret it as

$$\tilde{\mathbf{y}} = \mathbf{X}\mathbf{w}, \tag{1}$$

where vector $\tilde{\mathbf{y}}$ is a prediction of the target label vector $\mathbf{y}$ and term $\mathbf{X}$ is the data matrix with each row representing one shifted version of image sample $\mathbf{x}$. Given a number M of correlation filters, their predictions can be aggregated by a weighting process. Thus, Eq. (1) can be rewritten as

$$\tilde{\mathbf{y}} = \sum_{i=1}^{M} a_i \mathbf{X}_i \mathbf{w}_i \quad s.t. \quad \sum_{i=1}^{M} a_i = 1 \ \wedge \ a_i \geq 0 \tag{2}$$

with the weight vector $\mathbf{a} = [a_1, \ldots, a_M]$. Despite that the same image sample $\mathbf{x}$ is processed, the CFs may differ in the feature extraction and data manipulation procedure. Hence, we assign a separate data matrix $\mathbf{X}_i$ to each model i. The training of multiple CFs can be interpreted by minimizing the following loss function

$$L(\mathbf{w}_1, \ldots, \mathbf{w}_M, \mathbf{a}) = \|\mathbf{y} - \sum_{i=1}^{M} a_i \mathbf{X}_i \mathbf{w}_i\|^2 + \sum_{i=1}^{M} \lambda_i a_i \|\mathbf{w}_i\|^2, \tag{3}$$

where term λ_i is a non-negative regularization factor for model i. Without loss of generality, we proceed the discussion on a special case with only 2 models and thus the loss function only contains three parameters $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{a}$. Considering the fact that by fixing each two of these parameters, the loss function for the remaining parameter is in a quadratic form, which is convex,[1] thus problem (3) is verified as tri-convex with $M = 2$. Leveraging this property, we jointly learn the models by iteratively minimizing the loss with alternatively optimizing each of these three parameters, which is in analogy to the approach of Alternate Convex Search (ACS) in solving biconvex problems. Therefore, problem (3) can be decomposed into solving three convex subproblems and in each iteration, we proceed with following three steps:

[1]The convexity is inferred by the positive definite Hessian matrix, which is introduced in following parts. For simplicity, we omit the constant multiplier in front of the Hessian matrix, as it does not affect the convexity.

a) Updating filter coefficient $\mathbf{w}_1$. By fixing the coefficient vector of the second model $\mathbf{w}_2$ and the weighting vector $\mathbf{a} = [a_1, a_2]$, loss function (3) can be simplified as

$$L_1 = \|\mathbf{y}_1' - \mathbf{X}_1 \mathbf{w}_1'\|^2 + \lambda_1' \|\mathbf{w}_1'\|^2 \tag{4}$$

subject to modified parameters

$$\begin{cases} \mathbf{y}_1' = \mathbf{y} - a_2 \mathbf{X}_2 \mathbf{w}_2 \\ \mathbf{w}_1' = a_1 \mathbf{w}_1 \\ \lambda_1' = \lambda_1 / (a_1 + \epsilon) \end{cases} \tag{5}$$

where term ϵ is a small positive number to avoid devision by zero. Since both vectors $\mathbf{w}_2$ and $\mathbf{a}$ are constant, solving the coefficient vector $\mathbf{w}_1$ is equivalent to solving the new vector $\mathbf{w}_1'$. The Hessian matrix of L_1 is $\mathbf{X}_1^\top \mathbf{X}_1 + \lambda_1'$, where the first term is a Gram matrix which is positive semi-definite while the second term λ_1' is a positive number. Thus, the entire Hessian matrix is positive definite and L_1 is thereby convex. The subproblem (4) can be solved by just regarding it as a normal correlation filter and $\mathbf{w}_1'$ represents the filter coefficients. Due to the circular shifting of training samples in the data matrix $\mathbf{X_1}$, according to [16], $\mathbf{w}_1'$ can be solved in the frequency domain by reinterpreting Eq. (4) as $\hat{L}_1 = \|\hat{\mathbf{y}}_1' - \hat{\mathbf{x}}_1 \odot \hat{\mathbf{w}}_1'\|^2 + \lambda_1' \|\hat{\mathbf{w}}_1'\|^2$, where $\mathbf{x_1}$ is the original image sample, symbol $\hat{}$ indicates the Fourier transform and operator $\odot$ denotes the Hadamard product. The new loss function can be solved analytically, e.g., by the approach of [31]. Since the new label vector $\mathbf{y}_1'$ just represents the residuals between the true labels and the predicted ones by the second model, the first model is learned on $\mathbf{y}_1'$ so that the classification errors for the whole tracker can be reduced.

b) Updating filter coefficient $\mathbf{w}_2$. Analogously, by fixing $\mathbf{w}_1$ and $\mathbf{a}$, the loss function (3) can be reinterpreted as

$$L_2 = \|\mathbf{y}_2' - \mathbf{X}_2 \mathbf{w}_2'\|^2 + \lambda_2' \|\mathbf{w}_2'\|^2 \tag{6}$$

$$s.t. \quad \begin{cases} \mathbf{y}_2' = \mathbf{y} - a_1 \mathbf{X}_1 \mathbf{w}_1 \\ \mathbf{w}_2' = a_2 \mathbf{w}_2 \\ \lambda_2' = \lambda_2 / (a_2 + \epsilon) \end{cases} \tag{7}$$

Similar to step a), it can be demonstrated that L_2 is convex. Obviously, the second model is also learned on the prediction error of the first model, i.e., the new label vector $\mathbf{y}_2'$. Here we replace the multiplication between the coefficient vector $\mathbf{w}_2'$ and each shifted version of sample $\mathbf{x}_2$ (only differing from $\mathbf{x}$ in feature representation) with a convolution operation $*$ and thus equation (6) can be rewritten as

$$L_2 = L_w + \lambda_2' c_w \tag{8a}$$

$$s.t. \quad \begin{cases} L_w = \|\mathbf{y}_2' - \mathbf{w}_2' * \mathbf{x}_2\|^2 & \text{(8b)} \\ c_w = \|\mathbf{w}_2'\|^2 & \text{(8c)} \end{cases}$$

Since term (8) can be considered as the prediction error by one convolutional layer and term (8) is equivalent to the L_2-norm of weight decay, Eq. (8) is consistent with the general form of the CNN loss function for one sample [32]. Hence, the second filter can be constructed

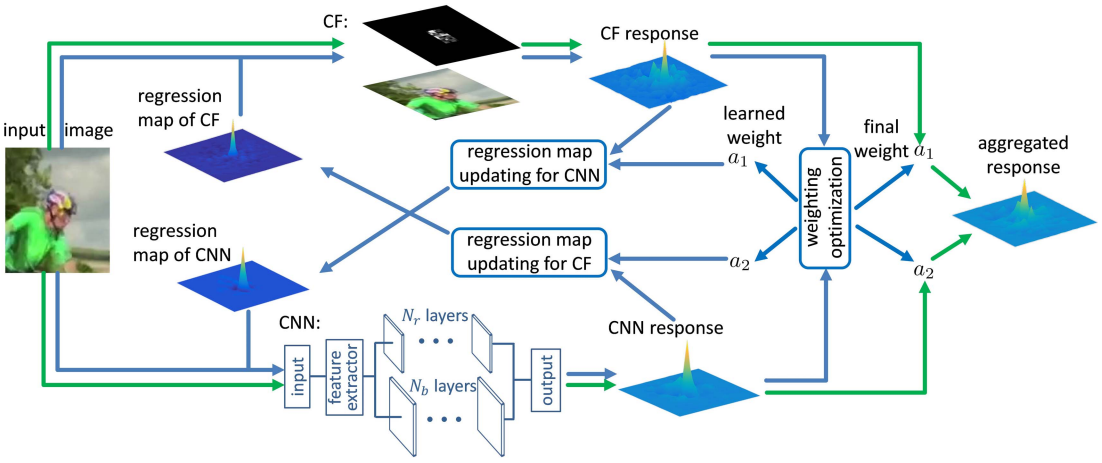


Fig. 1. Proposed collaborative learning architecture. In this approach, the CF and CNN model and their weights are learned by alternating optimization. Different from conventional approaches, we learn both models based on the errors of each other. The whole learning procedure is conducted iteratively and after each iteration, we update the regression map for each model. Their responses are aggregated for the final target inference. The training and inference process are respectively indicated by blue and green arrows.

in a CNN-form and $\mathbf{w}'_2$ represents the its filter coefficients. It can be iteratively optimized by the stochastic gradient descent algorithm, interpreted as $\mathbf{w}'_{2,k+1} = \mathbf{w}'_{2,k} - \rho \frac{\partial L_2}{\partial \mathbf{w}'_{2,k}}$, where k is the iteration index and ρ is the step size.

c) Updating weighting a. Substituting the learned filter $\mathbf{w}_1$ and $\mathbf{w}_2$ from previous steps into Eq. (3), it can be reformulated as a quadratic programming problem of loss function

$$L_a = \|\mathbf{y} - \mathbf{P}\mathbf{a}\|^2 + \mathbf{q}^\top \mathbf{a} \tag{9a}$$
$$= \mathbf{a}^\top \mathbf{P}^\top \mathbf{P}\mathbf{a} + (\mathbf{q}^\top - 2\mathbf{y}^\top \mathbf{P})\mathbf{a} + \mathbf{y}^\top \mathbf{y}$$
$$s.t. \ \sum a_i = 1 \ \wedge \ a_i \geq 0, \tag{9b}$$

where we denote $\mathbf{q} = [\lambda_1 \|\mathbf{w}_1\|^2, \lambda_2 \|\mathbf{w}_2\|^2]^\top = [q_1, q_2]^\top$ and $\mathbf{P} = [\mathbf{X}_1\mathbf{w}_1, \mathbf{X}_2\mathbf{w}_2] = [\mathbf{p}_1, \mathbf{p}_2]$. The Hessian matrix of L_a is $\mathbf{P}^\top\mathbf{P}$, which is a Gram matrix and thus positive semi-definite. To make problem L_a convex, matrix $\mathbf{P}^\top\mathbf{P}$ should be positive definite. It means that the matrix should contain positive eigenvalues $\lambda_{a,i}$ $(i = 1, 2)$, which are mathematically solved by $\det(\mathbf{P}^\top\mathbf{P} - \lambda_a \mathbf{I}) = 0$. This equation can be further reformulated as $\lambda_a^2 - (\mathbf{p}_1^\top \mathbf{p}_1 + \mathbf{p}_2^\top \mathbf{p}_2)\lambda_a + \det(\mathbf{P}^\top\mathbf{P}) = 0$. For positive eigenvalues, it requires the term $\mathbf{p}_1^\top \mathbf{p}_1 + \mathbf{p}_2^\top \mathbf{p}_2$ and the determinant $\det(\mathbf{P}^\top\mathbf{P})$ to be positive. Since the first term only contains quadratic elements and $\mathbf{P}^\top\mathbf{P}$ is a Gram matrix, both terms are already non-negative. The zero case of eigenvalue (i.e., $\det(\mathbf{P}^\top\mathbf{P}) = 0$) only occurs, when response maps $\mathbf{p}_1$, $\mathbf{p}_2$ are highly correlated or at least one of them equals zero. However, since classification models utilized in CF and CNN significantly differ from each other, their response maps are less correlated. In experiments, we also found that their filter coefficients are never zero. For the CF, the utilized features are a concatenation of the FHOG [33] and Color Attributes [17] while in the CNN tracker the VGG16 is utilized as a feature extractor. In experiments, their feature vectors as well as the responses are never zero. Thus, in our case, both terms $\mathbf{p}_1^\top \mathbf{p}_1 + \mathbf{p}_2^\top \mathbf{p}_2$ and $\det(\mathbf{P}^\top\mathbf{P})$ are positive. Consequently, the matrix $\mathbf{P}^\top\mathbf{P}$ only contains positive eigenvalues and is positive definite. Therefore, L_a is also convex.

According to constraint (9b), we can replace weight a_2 with $a_2 = 1 - a_1$. Introducing it into the loss (9b), we obtain a second order function with only one variable a_1 and it can be interpreted as a parabola (U-shaped). Depending on the position of the bottom point of the parabola, we provide following solution

$$a_1 = \begin{cases} 0, & \text{for} \quad 2c_2 + c_3 \geq 0 \\ 1, & \text{for} \quad 2c_2 + c_3 \leq -2c_1 \\ -\dfrac{2c_2 + c_3}{2c_1 + \epsilon}, & \text{otherwise} \end{cases} \tag{10}$$

with definitions of $c_1 = \|\mathbf{p}_1 - \mathbf{p}_2\|^2$, $c_2 = (\mathbf{p}_1 - \mathbf{p}_2)^\top (\mathbf{p}_2 - \mathbf{y})$ and $c_3 = q_1 - q_2$. The first two cases respectively indicate that the bottom point of the parabola is on the left/right outside of the range $0 \leq a_1 \leq 1$ while in the third case, the bottom point is located within this range. The second weight element is easily obtained by $a_2 = 1 - a_1$. With above analytical form, the weighting vector $\mathbf{a}$ can be computed efficiently.

In such alternating optimization fashion, we can incorporate both CF and CNN based models into one joint learning architecture, as illustrated in Fig. 1. In the first iteration, filter coefficients are initialized by separately training each model alone and further substituted into step c) to initialize the weight vector $\mathbf{a}$. For processing efficiency, in our experiments, the learning process is terminated after N_t iterations.

B. Classification Models

1) Implementation of the Correlation Filter: A recent study [21] has shown that training CF on extended image region can significantly boost the classification performance. Thus, we employ the same concept for our correlation filter. Furthermore, the learning of CNN model also benefits, because

it processes the same image region and more samples are involved. Here we interpret the loss function of our CF model as

$$L_{CF} = \|\mathbf{y} - \mathbf{X}\mathbf{g}\|^2 + \lambda\|\mathbf{w}\|^2 \quad s.t. \quad \mathbf{g} = \mathbf{\Gamma}\mathbf{w}, \tag{11}$$

where term $\mathbf{\Gamma}$ is a zero padding operator, just to ensure that sample $\mathbf{x}$ is extracted from an enlarged image region while coefficient vector $\mathbf{w}$ shares the same size of the target. Please note that, for generalization purpose, we omit the subscripts for parameters $\mathbf{X}$, $\mathbf{w}$ and λ. For clarity, we constrain the following discussion of this problem to its spatial form, although it can be efficiently solved in the frequency domain. According to the Augmented Lagrangian Method (ALM) [34], loss function (11) can be reformulated as

$$L(\mathbf{g}, \mathbf{w}, \boldsymbol{\zeta}) = \frac{1}{2}L_{CF} + \boldsymbol{\zeta}^\top(\mathbf{g} - \mathbf{\Gamma}\mathbf{w}) + \frac{\mu}{2}\|\mathbf{g} - \mathbf{\Gamma}\mathbf{w}\|^2 \tag{12}$$

with a penalty factor μ and the Lagrangian vector $\boldsymbol{\zeta}$. To solve this problem, we employ the accelerated Alternating Direction Method of Multipliers (ADMM) with restart operation [31]. Comparing with the conventional ADMM approach, the main idea of utilized algorithm is that the optimization step will be reset and other parameters roll back to their last states if the combination of primal and dual error does not decrease by a factor of at least η. This ensures the monotonicity of the residual and thus a faster convergence rate can be achieved. Empirically, the scaling factor of error term is set to $\eta = 0.999$, as recommended in work [31]. We also refer readers to this work for details about the algorithm.

2) Implementation of the Convolutional Neural Network: According to the signal processing theory, a big filter can be decomposed into a convolution of several smaller filters and filtering results are subject to linear operation. Therefore, it enables our model to deploy N_b convolutional layers (denoted as the base network). To alleviate the degradation effect by a deep net, we also employ the residual learning concept [35] by adding N_r additional convolutional layers to learn the residual between the outputs of base network and the target labels. The whole structure of our CNN model is illustrated at the bottom of Fig. 1. In our experiments, the filter size of each residual layer is defined as 1×1 pixel while for base layers, their filter sizes are adjusted according to the equation

$$s_{obj} \approx \sum_{i=1}^{N_b} s_{b,i}, \tag{13}$$

where s_{obj} indicates the height or width of the tracked object and $s_{b,i}$ denotes the height or width of filter i. Eq. (13) ensures that information from the whole target can contribute to the final response. To simplify the implementation, we employ the same filter size for all base layers and thus we have $s_{b,i} = \frac{s_{obj}}{N_b}$. Additionally, we employ the ReLU unit as the activation function for each layer. Since the correlation filter is learned on circularly shifted samples, for consistence, we conduct circular padding for the input image during training the CNN model.

C. Tracking and Updating

Since our architecture employs both CF and CNN based trackers, for target inference, their response maps are aggregated according to corresponding weights, interpreted by Eq. (2). As their map sizes may differ from each other, we perform a cubic interpolation so that the output of our CNN model shares the same size with the response map of correlation filter. For dynamically adapting to the current object appearance, after the target is identified, we firstly update the object appearance by the equation

$$\mathbf{x}^t = (1 - \psi)\mathbf{x}^{t-1} + \psi\mathbf{x}^{cur}, \tag{14}$$

where $\mathbf{x}^{t-1}$ denotes the training sample from time $t-1$. Term $\mathbf{x}^{cur}$ represents the sample extracted from current image and ψ is a small learning rate (in experiments it is empirically set to 0.016). The updated image patch will be fed into our joint learning framework to retrain both the CF and CNN tracker. We follow the alternate optimization scheme introduced before. The weights are directly calculated by the third step introduced in Section III-A. Note that the update is conducted in feature level, due to different features utilized, we update both models separately. To cope with scale variation, we resize the tested image region in 5 scales with a scaling factor of 1.05. On the top of Fig. 2, we plot the weight of both models for the Biker-sequence from the OTB dataset [10]. In frame 27, the biker moves close to the camera. As the appearance merely changes, both CF and CNN models enjoy nearly equal weights. In frame 89, he is jumping to the other side and the tracked head suffers from great pose variation. As deep features are proven more robust, the CNN model is strongly weighted in this case. At the bottom of Fig. 2, we present the weight curve in dependence of the iteration number N_t. Obviously, in both cases, the weights converge after 2 iterations. In our experiments, similar behaviors can also be found in other frames as well as in other test sequences. Thus, the iteration number of alternating optimization is set to $N_t = 2$.

IV. Experiments and Results

A. Experimental Setup

We evaluate our architecture on three standard visual tracking benchmarks: OTB2013 [10], OTB100 [2] and VOT16 [36]. The first dataset consists of 50 sequences of various object classes (including pedestrians and vehicles) captured under circumstances like varied illumination, fast motion, rotation, etc. The second dataset increases the first one with 50 more sequences, thus it is considered more challenging. The last benchmark consists of 60 sequences captured such as with occlusion, size variation and motion challenges but it emphasizes the accuracy and robustness of tested approaches. For the comparison set, we choose 16 state-of-the-art trackers, i.e., DeepSRDCF [8], BACF [21], SINT [27], siamfc3s [26], staple [20], MUSTer [19], HDT [23], LCT [37], MEEM [38], DSST [39], siamfc-r [26], SSKCF [40], DNT [36], MDNet_N [24], HCF [37] and LGT [41]. Here we directly use their codes or results provided by their original papers. Since these trackers are

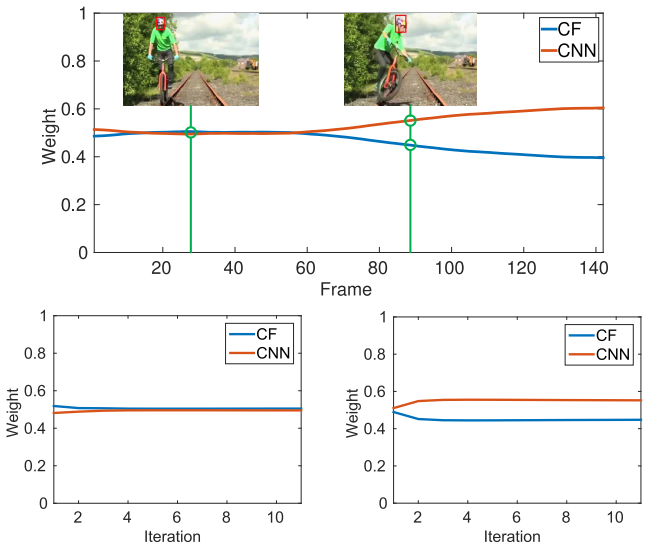


Fig. 2. In the first row, we plot the weights of CF and CNN models for the Biker-sequence from the OTB dataset. Here we choose frame 27 and frame 89 as examples and mark the corresponding points in green. In the second row, for each selected frame, we respectively show the weight curves in dependence of the iteration number N_t.

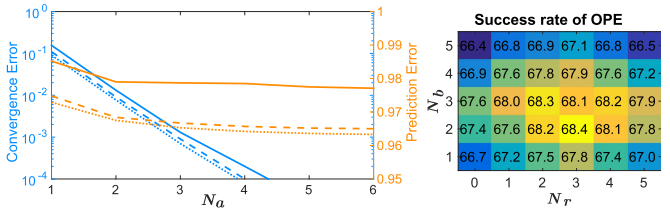


Fig. 3. On the left side, we plot both convergence and prediction error in dependence of iteration N_a in ADMM and respectively denote them in blue and orange. Both errors are calculated with the L_2-norm in spatial domain and averaged over all frames of the tested sequence. For each error type, we compare two ADMM approaches: the conventional one and the accelerated version, respectively denoted in solid and dash lines. On the right are success rates of our approach on the dataset OTB2013 in dependence of the base layer number N_b and the residual layer number N_r, displayed in heat map.

not tested on all the utilized datasets, we only compare them on the datasets where their results are available. For our CF model, we choose FHOG [33] and Color Attributes [17] as features. Other settings of our CF model are same as the BACF tracker. For our CNN model, we choose features from the $conv4-3$ layer of the VGG16 net[2] and shrink their channels to 32 by the PCA method. Thereafter, the deep features will be fed into both base and residual layers, whose outputs are summed and then given into the L_2 regression loss layer to predict the labels. For each sequence, our tracker is retrained on the first frame with the given bounding box. The training of CF is the same as described in Section III-B.1 and takes about 30 ms. For the CNN model, its first part (i.e., the VGG net) is only utilized as a feature extractor. Thus, it is kept unchanged and only the second part (i.e., the small residual net) will be retrained. At the first frame, the epoch number is set to 30, which yields a delay of about 100 ms. In the following frame, the CF is updated according to Eq. (14). The residual net in the CNN model is updated by the current sample with 2 epochs (which is found sufficient and only takes about 20 ms). The learning rate for CNN is set to 10^{-6}. The CF is implemented by Matlab and runs on CPU (Intel i7-6700 with a memory of 16GB) while the CNN is implemented by the MatConvNet library and runs on the GPU of NVIDIA TITAN Xp.

B. Results and Evaluation

1) Evaluation on Key Factors of CF Model: Our CF model is trained by the ADMM approach, which optimizes iteratively. Thus, the iteration number N_a should be well selected in terms of both classification power and processing efficiency. Here we still take the Biker-sequence from the OTB dataset as an example. On the left side of Fig. 3, we plot the convergence error of filter coefficient and the error of total predicted labels in dependence of iteration N_a and respectively denote them in blue and orange. Both of these errors are calculated with the L_2-norm in spatial domain and averaged over all frames of the sequence. Additionally, in this plot we compare two ADMM approaches: the conventional one and the accelerated version, respectively denoted in solid and dash lines. Obviously, the accelerated ADMM enjoys a faster convergence and a slightly better prediction error, which verifies its effectiveness. As the prediction error converges after two iterations (similar behavior is also found in other sequences), we set $N_a = 2$ for further experiments.

[2]The net utilized in this work is pre-trained on the ImageNet and provided by the MatConvNet libraries for Matlab.

TABLE I

COMPARISON OF DIFFERENT VERSIONS OF ADMM WITH FIXED ITERATION NUMBER. "accADMM" INDICATES THAT THE ACCELERATED ADMM IS INTEGRATED

	Success rate	Runtime (ms)	Convergence error
ADMM	0.678	28.5	0.014
accADMM	0.681	30.4	0.009

TABLE II

COMPARISON OF DIFFERENT VERSIONS OF ADMM WITH FIXED THRESHOLD OF CONVERGENCE ERROR. "accADMM" INDICATES THAT THE ACCELERATED ADMM IS INTEGRATED

	Success rate	Runtime (ms)	Average iterations
ADMM	0.680	35.2	3.42
accADMM	0.681	32.3	2.68

2) Evaluation on the Fast ADMM Solver: For a more quantitative impression, we provide two test results on the OTB2013 benchmark. For both tests, we compare two trackers: BACF integrated with the naive ADMM and BACF integrated with the accelerated ADMM. We set the same parameters for both trackers. In the first test, for both versions of ADMM, the maximum iteration number is set to 2. We report the success rate, the average runtime (in milliseconds) and the average convergence error in Table I. The calculation of success rate is consistent with the work [10]. In the second test, we remove the limit on iterations but set the threshold of convergence error to 1e-3. Accordingly, we record the success rate, the average runtime and the average iterations in Table II.

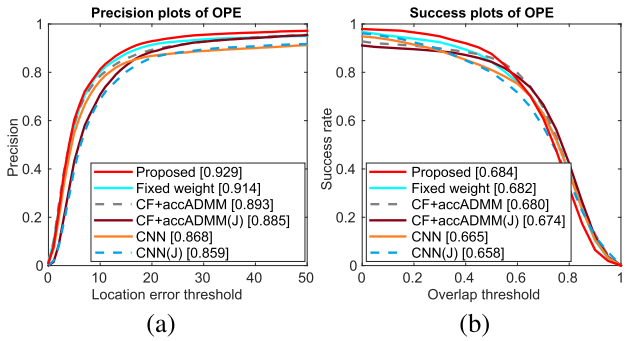


Fig. 4. Evaluating individual trackers on the OTB benchmark.

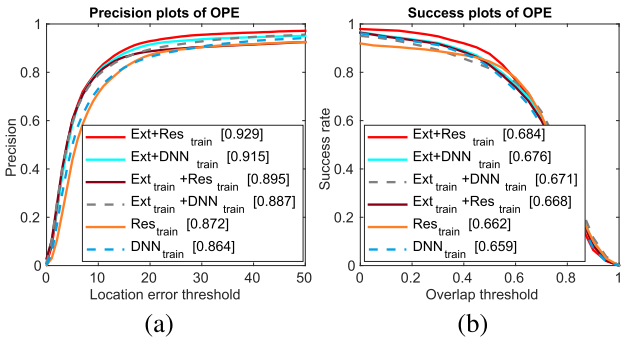


Fig. 5. Evaluating different CNN learning strategies on the OTB benchmark.

Apparently, with the same iterations, the accelerated ADMM achieves a smaller convergence error (Table I), which means the solution is more precise. Thus, it achieves a better tracking performance. The cost is a slight increase in the processing time, which is due to the additional reset and rolling back operations. With the same threshold of convergence error, the accelerated ADMM needs less iterations (Table II), which further proves its efficiency.

3) Evaluation on Key Factors of CNN Model: Although we have introduced our CNN model in last section, there is still one question unanswered, i.e., how to choose the layer number for both base and residual network in our CNN model. Here we determine these parameters based on experiments on the OTB2013 dataset. We measure our tracking accuracy by the area-under-curve (AUC) score in terms of success plot according to the One Pass Evaluation (OPE) [10]. In this protocol, it triggers the tracker with the first bounding box and records the estimated object states in following frames. The test results are reported in the heat map on the right side of Fig. 3. Accordingly, the highest value is located at the point of $(N_b = 2, N_r = 3)$ and it decreases for more layers, which is due to the overfitting by deep net structure. Thus, we choose this point as the optimal setting in our further experiments.

4) Evaluation of Individual Components: For a more clear impression about the performance of utilized individual trackers, we provide an additional experiment. Here we evaluate six cases: 1) The proposed joint tracker; 2) Separately trained CF and CNN tracker, which are aggregated with weights calculated based on their performance by Eq. (9); 3) The CF tracker only (using accelerated ADMM solver); 4) The CNN tracker only; 5) The jointly trained CF and 6) CNN tracker. The evaluation results are reported by the precision and success plot on the OTB2013 benchmark, illustrated in Fig. 4. We follow the OPE protocol and rank the trackers w.r.t. the location error of 20 pixels and the AUC score. The individual trackers trained by our joint learning approach are denoted with the suffix "(J)". It is clear that the CF tracker outperforms the CNN tracker. But both of them are inferior to the combined method with fixed weights. It implies that CF and CNN emphasize different image information, which is somehow complementary to each other. Due to the optimization of both models and their weights, the proposed approach outmatches above methods on both precision and success rate, which proves that our approach is more effective to combine the classification power of different models. Considering the individual trackers trained by our joint learning approach, they only yield an inferior performance to those by normal training procedures. We assume that by the joint learning both CF and CNN focus more on specific image information, which is somehow complementary. Although their individual performance cannot benefit from the joint learning, their aggregated classification power improves significantly in the proposed approach.

5) Evaluation of CNN Learning Strategies: Here we provide an additional experiment to explore the influence of different learning strategies for the CNN tracker. In this experiment, we compare 3 cases: 1) The proposed approach, i.e., the VGG16 as a feature extractor and only online training the residual net, which is denoted as "Ext+$\text{Res}_{\text{train}}$"; 2) Online training of both the feature extractor and the residual net, denoted as "$\text{Ext}_{\text{train}}$+$\text{Res}_{\text{train}}$"; 3) Removing the feature extractor and directly training the residual net, denoted as "$\text{Res}_{\text{train}}$". In the last case we adapt the input layer, so that the image can be directly fed into the network. Furthermore, we replace the residual net with a fully connected network (denoted as DNN), and repeat the above configurations, which yields the another 3 comparison sets: 4) "Ext+$\text{DNN}_{\text{train}}$", 5) "$\text{Ext}_{\text{train}}$+$\text{DNN}_{\text{train}}$" and 6) "$\text{DNN}_{\text{train}}$". With the last three comparison sets, we explore the performance of different network structures. For a fair comparison, the layer number of DNN is same as that of the residual net. In each DNN layer, we also adjust its neuron numbers so that the number of neuron connections is equivalent to that in the residual net. We use the same hyper-parameters to learn each network. All the above networks run on GPU. The test results on the OTB2013 benchmark are reported in Fig. 5.

It is clear that the configuration "Ext+$\text{Res}_{\text{train}}$" employed by the proposed approach performs best in terms of both precision and success rate. By replacing the utilized residual net with a fully connected net (DNN), the precision and success rate are slightly reduced by about 1% in "Ext+$\text{DNN}_{\text{train}}$". We assume that the property of translation invariance brought by the convolutional kernels in the residual net is more advantageous for target classification in our case. If the feature extractor (i.e., VGG16) is also trained online, the precision and the success rate are further reduced in "$\text{Ext}_{\text{train}}$+$\text{Res}_{\text{train}}$", respectively by about 3% and 2%. Since the VGG16 is pretrained on the ImageNet and already optimized for general

TABLE III
AVERAGE PROCESSING TIME OF EACH CNN LEARNING STRATEGY

Time (s)	Ext+$\text{Res}_{\text{train}}$	$\text{Ext}_{\text{train}}$+$\text{Res}_{\text{train}}$	$\text{Res}_{\text{train}}$	Ext+$\text{DNN}_{\text{train}}$	$\text{Ext}_{\text{train}}$+$\text{DNN}_{\text{train}}$	$\text{DNN}_{\text{train}}$
feature extractor	0.011	0.011	-	0.011	0.011	-
learning	0.021	0.087	0.021	0.019	0.081	0.019
inference	0.005	0.005	0.005	0.004	0.004	0.004
total	0.037	0.103	0.026	0.034	0.096	0.023

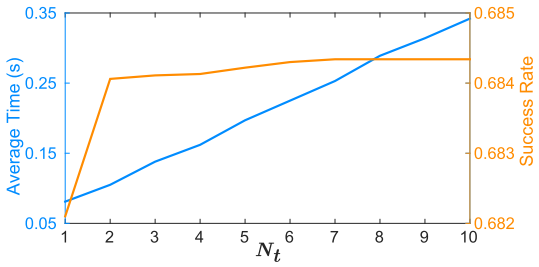


Fig. 6. The average time cost for each frame and the success rate in terms of iteration number N_t. Experiment results are from the OTB2013 dataset. The success rate is measured by the AUC score according to the OPE protocol [10].

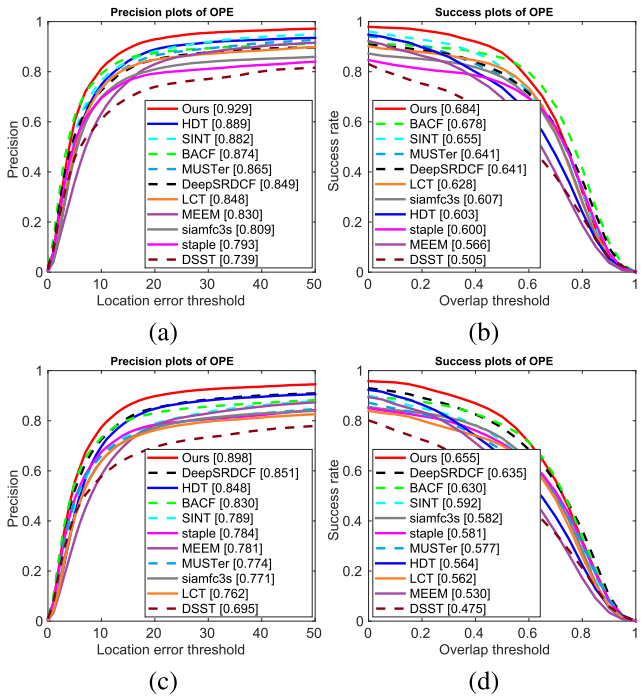


Fig. 7. Tracking results of compared approaches on datasets OTB2013 and OTB100 are respectively presented in the first and the second row. The precision and success plot are respectively presented in the first and second column.

classification tasks, the retraining with very limited samples and iterations may reduce its discriminative power. Without the feature extractor in "$\text{Res}_{\text{train}}$", both the precision and success rate decrease significantly, which is due to the fact that a small network is not able to encode powerful features as the VGG16.

Additionally, we report the average time cost of each network especially for learning and inference in Table III. The total time cost for the CNN tracker utilized in the proposed approach is 37 milliseconds. By training the feature extractor online, the time cost is nearly tripled. Without employing the feature extractor, nearly one third of the processing time can be reduced, but at the expense of significantly decreased accuracy. By replacing the residual net with a DNN, it only brings minor boosting of the processing efficiency.

6) Evaluation on Termination Condition: To reveal the influence of the termination condition of our optimization procedure on the tracking performance, we conducted another experiment on the OTB2013 benchmark. Both the average time cost for each frame and the tracking success rate in terms of iteration number N_t are illustrated in Fig. 6. The success rate is measured by the AUC score according to the OPE protocol [10]. From the results it can be seen that the success rate settles down after 2 iterations while the average time cost increases proportional to the iteration number N_t. As more iterations can only aggravate the computational burden and cannot further improve the tracking performance, the point of $N_t = 2$ is adopted as the optimal setting in our experiments. This is also consistent with the example introduced in Section III-C.

7) Evaluation on OTB Datasets: For evaluation on dataset OTB2013 and OTB100, we strictly follow the OPE-protocol [10]. The tracking results are presented with plots of success rate and precision, illustrated in Fig. 7. The first plot type is calculated in terms of the overlap between predicted object and its ground truth and all trackers are ranked according to the AUC score. The second plot type is calculated in terms of the location error and the ranking is made at the threshold of 20 pixels. According to the results on OTB2013 (first row in Fig. 7), our approach performs best for both metrics. In the success plot, we slightly outperform the second best tracker, i.e., the BACF (same as our CF model), by 0.6%. We owe such improvement to the CNN model incorporated in our architecture. In the precision plot, the gain becomes more obvious, more than 5%, implying that the inferred location by the deep model is more precise. As the OTB100 dataset is more difficult, results of all tested trackers decrease for both metrics (second row in Fig. 7). Still, our approach is on the top. On the second place is the DeepSRDCF tracker, which trains a CF directly on deep features from an enlarged image region. It can be considered as a sparse cooperation between CF and CNN. Comparing with it, our approach respectively achieves a gain of 4.7% and 2% in each metric, which verifies the effectiveness of our joint learning architecture.

8) Attribute Based Evaluation: In the next experiment, which is based on sequence attributes provided by the OTB benchmark, we respectively evaluate the performance of our architecture in eight challenging scenarios including background clutter, illumination variation, occlusion, deformation, fast motion, in-plane rotation, out-of-plane rotation and motion blur. Test results of compared methods are presented by plots

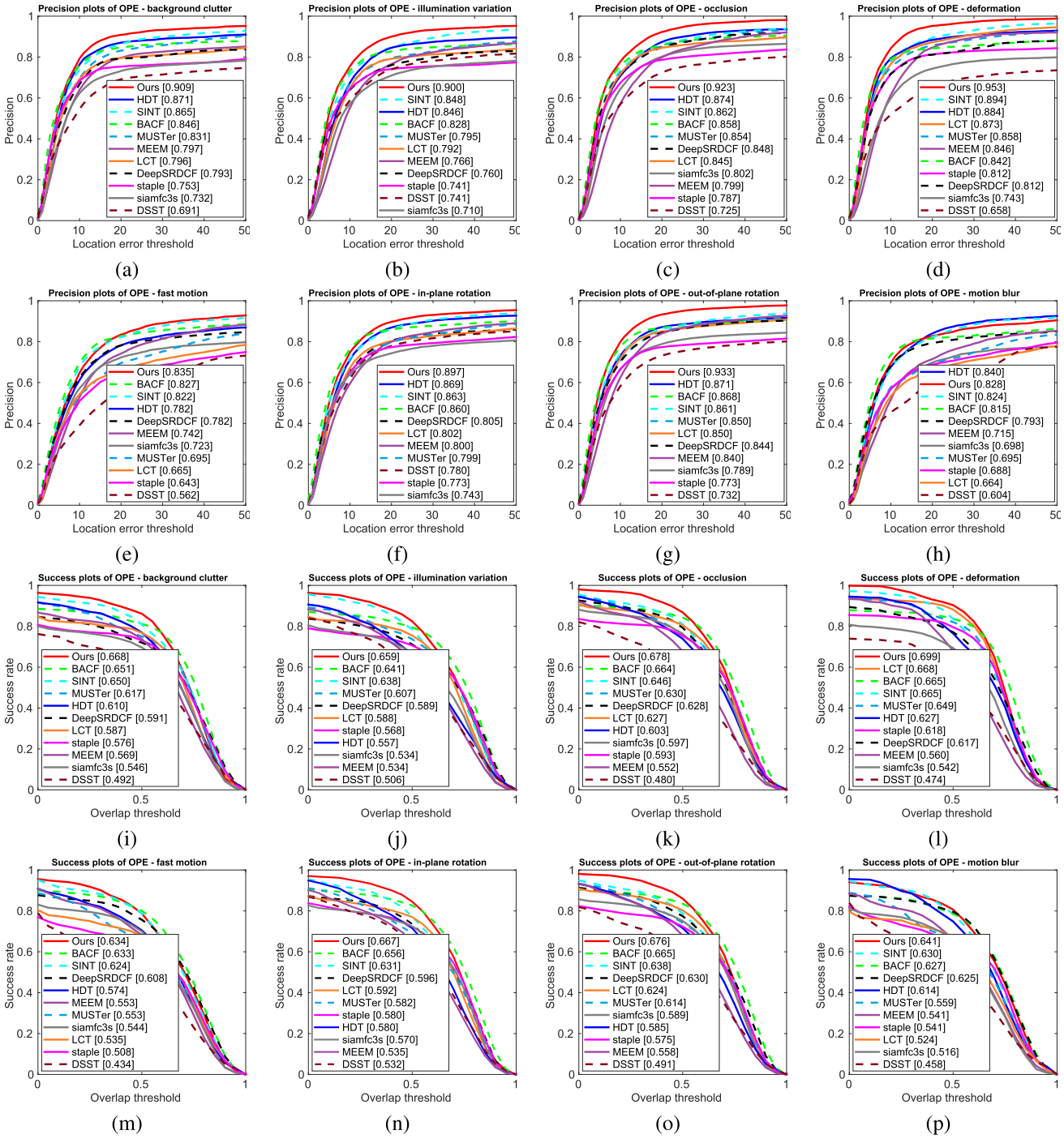


Fig. 8. Tracking results for each attribute. Precision curves are presented in the first two rows and below them are the corresponding success plots.

of both precision and success rate in Fig. 8. It is clear to be seen that in 7 out of 8 attribute tests, our approach is ranked at the first place in terms of both metrics. Especially for attributes like background clutter and illumination variation, the gain of our tracker is about 2% to 5% in both metrics, proving that our collaborative architecture is more robust in these cases. For other attribute, e.g., the fast motion, our gain especially in terms of success rate is not significant, only 0.1% over the BACF tracker. We assume that this attribute is more relevant to image searching strategies rather than the utilized models. As features extracted from blurred images strongly affect the classification performance of both CF and CNN models, our approach performs inferior to the HDT tracker by about 1% in terms of precision value. However, we still surpass it by a higher success rate, which implies that estimated target size by our tracker is more accurate. For a qualitative impression, we provide several tracking examples in Fig. 9.

9) Evaluation on VOT16 Benchmark: We provide the test results on VOT16 in Table IV. Here we totally compare

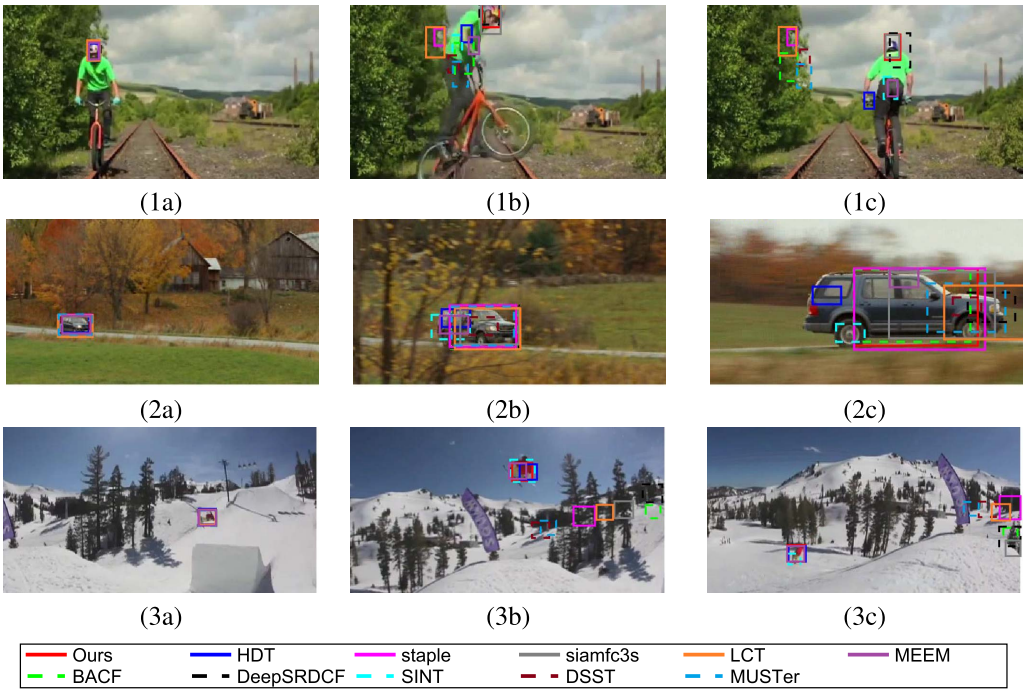


Fig. 9. Tracking examples with attributes of fast motion, perspective variation, rotation, etc. Each row represents one sequence.

TABLE IV

EVALUATION ON VOT16 W.R.T. ACCURACY (A), ROBUSTNESS RAW VALUE (R), AVERAGE OVERLAP (AO) AND EXPECTED AVERAGE OVERLAP (EAO). SYMBOL ↑ DENOTES THAT HIGHER VALUES ARE BETTER, WHILE ↓ PREFERS LOWER VALUES. WE MARK THE TOP METHOD IN BOLD

Metrics	DeepSRDCF	SSKCF	siamfc-r	DNT	MDNet_N	HCF	LGT	DSST	Ours
A (↑)	0.528	0.547	**0.549**	0.515	0.541	0.450	0.420	0.533	**0.549**
R (↓)	0.326	0.373	0.382	0.329	0.337	0.396	0.605	0.704	**0.309**
AO (↑)	0.427	0.391	0.421	0.427	**0.457**	0.374	0.271	0.325	0.448
EAO (↑)	0.276	0.277	0.277	0.278	0.257	0.220	0.168	0.181	**0.292**
speed (fps ↑)	<1	**30**	8	5	1	11	4	24	9.5

9 trackers w.r.t. the metrics of accuracy (A), robustness raw value (R), average overlap (AO) and expected average overlap (EAO). The accuracy measures the average overlap ratio between the estimated target bounding box and its ground truth. The raw robustness is calculated by average failure rates. The last two metrics are calculated based on the average overlap of a tracker on several short sequences and the main difference is that failure cases are specifically addressed in EAO. Except the robustness, high values are preferred by all other metrics. Note that we did not exactly use the same trackers as in previous experiments, because the results of some trackers on VOT16 are unavailable. From the table, it is obvious that our tracker performs best in terms of robustness raw value and accuracy, which further proves that our collaborative architecture is immune to challenges such as occlusions, illumination variation and fast motion. In terms of the AO metric, we rank on the second place while on the top is the MDNet_N tracker. This can be ascribed to the net structure itself. The MDNet_N adopts two networks with each consisting of 5 layers and several branches while our CNN model is only constructed with 2 base layers. Although such simple model is utilized in our framework, the gap between our method and the MDNet_N is minor, which is 0.9%. Among the compared approaches, MDNet_N, siamfc-r and DNT are neural network based trackers while others mainly rely on correlation filters. Since our tracker outperforms most of the CF and CNN based trackers, it further demonstrates the merit of our joint learning architecture which is able to reduce the classification error for the individual models.

10) Evaluation on Other Base Trackers: In previous parts of the paper, we already mentioned that our joint learning scheme is not limited to specific trackers. For validation of such proposition, here we provide two more experiments, in which we respectively replace our CF and CNN tracker with other trackers which are introduced in Section IV B 7) and open-source. The new combinations are tested on the OTB2013 benchmark and their precision scores are measured according to the OPE protocol [10] and reported in Table V. According to the results, it can be seen that by applying our joint learning scheme, the performance of all trackers have been boosted. Especially, in combination with CF, HDT achieves a comparable score with our approach. Since staple, MEEM, LCT and DSST employ similar handcrafted features as our CF, less complementary information can be provided. Thus, their combination with the CNN tracker shows a better performance than with the CF tracker. However, compared with their naive approaches, the precision increases with both combination types, which proves the effectiveness of our joint learning scheme even on weak trackers.

11) Runtime Performance: Among the state-of-the-art trackers in Table IV, the fastest one is the SSKCF which runs

TABLE V

PRECISION OF TRACKERS COMBINED WITH JOINT LEARNING SCHEME. +CF AND +CNN RESPECTIVELY DENOTES THAT THE NAIVE TRACKER IS COMBINED WITH OUR CF OR CNN TRACKER

	DeepSRDCF	HDT	SINT	staple	MEEM	LCT	DSST
naive	0.849	0.889	0.882	0.793	0.830	0.848	0.739
+CF	0.876	0.922	0.914	0.856	0.867	0.872	0.835
+CNN	0.882	0.905	0.891	0.864	0.872	0.885	0.858

TABLE VI

AVERAGE TIME COST OF EACH PROCEDURE IN THE PROPOSED APPROACH

	feature extraction		learning			inference		**total**
	CF	CNN	CF	CNN	weights	CF	CNN	
time (s)	0.016	0.011	0.034	0.021	0.002	0.016	0.005	0.105

at 30 frames per second (fps). And it is followed by the DSST tracker with a small gap of 6 fps while all the others fall in the speed range of less than 20 fps. Another phenomenon to be seen is that most of the well performed neural network based trackers can not run in real time, e.g., the DeepSRDCF and MDNet_N. Although our framework also utilizes CNN model, as its net structure is relative simple, we can still achieve a speed of 9.5 fps, which is sufficient for nearly real time applications. In our approach, most of the computations are conducted in training of both models, which takes more than half of the average time cost, as illustrated in Table VI. To train CF, according to [21], the computational complexity is bounded by $\mathcal{O}(NK\log(N))$, where N and K respectively denote the size (i.e., width multiplied by height) and the channel number of the feature map. For CNN, the complexity for one convolutional layer is interpreted as $\mathcal{O}(NKN_FK_F)$ with N_F and K_F respectively denoting the filter size and the number of its output channels. Considering our residual net, the filters for base layers are with equal sizes, which are comparable to the feature map. In our experimental setting, feature maps fed into the CF and the residual net are also with similar dimensions. Thus, the computational complexity of the multi-layer CNN is significantly higher than CF. However, since the CNN runs on GPU, its computation is highly parallelized. Thus, the time cost of CNN in experiments is not so high. Therefore, the bottleneck for runtime performance is mainly from CF (running on CPU). Since the proposed approach is implemented in Matlab and runs in a single thread, a further speed-up can be achieved by reimplementation in C++ and utilizing multi-threads, which is part of our future work.

12) Failure Case Analysis: Two examples of tracking failure are displayed in Fig. 10. In the first sequence (1a)-(1b), it tracks a bird which is flying into the cloud and disappeared for a long time. Since only a simple updating mechanism (e.g., the constant learning rate in CF) is employed in our approach, the classification model is contaminated by the corrupted samples during the disappearance of the target.

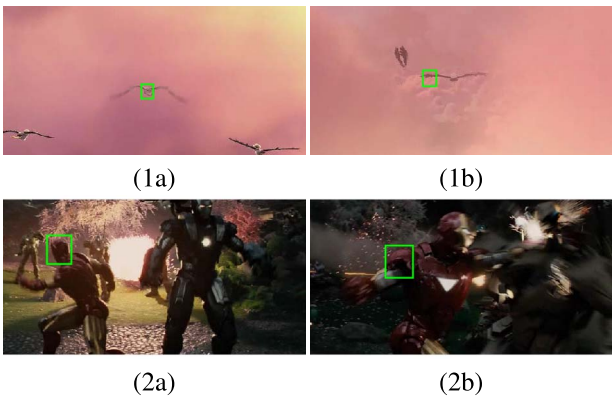


Fig. 10. Examples of tracking failures.

Therefore, after the bird flies out of the cloud with changed pose and position, it can not be correctly reidentified by our approach. In the second sequence (2a)-(2b), it tracks the head of the Ironman. Due to the very similar color and texture in his upper body, it is difficult for our tracker to distinguish between them and thus the tracker fails. Possible solutions to above failure cases are integrating redetection strategies to detect the target after its re-appearing and correctly estimating the motion of the target, e.g., by the optical flow, which are part of our future work.

V. CONCLUSION

In this paper, we proposed a novel collaborative visual tracking architecture, which is able to jointly learn both CF and CNN models. Such topic is essential for monitoring tasks of intelligent transportation systems yet has been less explored in current research works. The basic idea of our architecture is an alternating optimization approach in combination with a fast CF solver and an analytical solution for model weighting. In this architecture, both models are learned on the classification errors of each other and thus are able to persistently boost the discriminative power of the whole tracker. Although we only deploy relative simple CF and CNN models in our approach, we successfully demonstrated that the tracking performance can be significantly improved by our joint learning framework. Through experiments on standard benchmarks with various object classes, our architecture is demonstrated superior over most state-of-the-art approaches and robust in dealing with challenging scenarios. As our approach is not limited to the utilized models, it also enjoys a good generalization ability. As a part of our future work, we will extend our framework to more complex models and expect a further gain of the tracking performance.

REFERENCES

[1] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 263–270.

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

[3] W. Tian, L. Chen, K. Zou, and M. Lauer, "Vehicle tracking at nighttime by kernelized experts with channel-wise and temporal reliability estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 10, pp. 3159–3169, Oct. 2018.

[4] W. Tian and M. Lauer, "Tracking objects with severe occlusion by adaptive part filter modeling—In traffic scenes and beyond," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 4, pp. 60–73, 2018.

[5] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1, 2013, pp. 809–817.

[6] H. Li, Y. Li, and F. Porikli, "DeepTrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1834–1848, Apr. 2016.

[7] M. Kristan and A. Leonardis, "The thermal infrared visual object tracking VOT-TIR2016 challenge results," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2016, pp. 191–217.

[8] 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.

[9] Y. Li, Z. Xu, and J. Zhu, "CFNN: Correlation filter neural network for visual object tracking," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 2017, pp. 2222–2229.

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

[11] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2002, pp. 661–675.

[12] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, May 2008.

[13] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, vol. 1, no. 5, p. 6, Sep. 2006.

[14] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV)*, Sep. 2009, pp. 1393–1400.

[15] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "PROST: Parallel robust online simple tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 723–730.

[16] 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.

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

[18] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2015, pp. 254–265.

[19] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "MUltI-store tracker (MUSTer): A cognitive psychology inspired approach to object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 749–758.

[20] 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.

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

[22] Y. Li, Y. Zhang, Y. Xu, J. Wang, and Z. Miao, "Robust scale adaptive kernel correlation filter tracker with hierarchical convolutional features," *IEEE Signal Process. Lett.*, vol. 23, no. 8, pp. 1136–1140, Aug. 2016.

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

[24] 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.

[25] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 749–765.

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

[27] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1420–1429.

[28] K. Li, Y. Kong, and Y. Fu, "Multi-stream deep similarity learning networks for visual tracking," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 2017, pp. 1–7.

[29] 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.

[30] 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.

[31] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.

[32] M. A. Nielsen, *Neural Networks and Deep Learning*. San Francisco, CA, USA: Determination Press, 2015.

[33] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[34] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

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

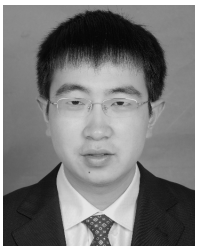
[37] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3074–3082.

[38] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 188–203.

[39] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2014, pp. 1–11.

[40] J.-D. Lee, Y.-S. Chen, and J.-C. Chien, "Classification and tracking of large vehicles for night driving," in *Proc. IEEE 5th Global Conf. Consum. Electron.*, Oct. 2016, pp. 1–2.

[41] L. Cehovin, M. Kristan, and A. Leonardis, "Robust visual tracking using an adaptive coupled-layer visual model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 4, pp. 941–953, Apr. 2013.

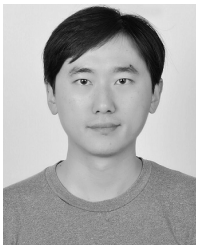


Wei Tian received the B.Sc. degree in mechatronics engineering from Tongji University, Shanghai, China, in 2010, and the M.Sc. degree in electrical engineering and information technology and the Ph.D. degree from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in May 2013 and in January 2019, respectively.

Since October 2013, he has been with the Institute of Measurement and Control Systems, KIT. He is currently working on computer vision and machine learning with research areas of robust object detection and tracking.

Niels Ole Salscheider received the M.Sc. degree in electrical engineering from RWTH Aachen University in 2014. He is currently pursuing the Ph.D. degree with the Institute of Measurement and Control Systems, Karlsruhe Institute of Technology. He is currently a Research Scientist with the Mobile Perception Systems Department, FZI Research Center for Information Technology. His research interests include machine learning and vision-based environment perception for automated driving.

Yunxiao Shan received the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 2018. From October 2015 to October 2016, he was a co-trained Ph.D. Student with the Rutgers University of American. He currently holds a post-doctoral position with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His areas of interests include the autonomous driving, mobile robots, and autonomous surface vehicles.



Long Chen received the B.Sc. degree in communication engineering and the Ph.D. degree in signal and information processing from Wuhan University, Wuhan, China, in 2007 and 2013, respectively. From October 2010 to November 2012, he was a co-trained Ph.D. Student with the National University of Singapore. From 2008 to 2013, he was an in-charge of environmental perception system for autonomous vehicle SmartV-II with the Intelligent Vehicle Group, Wuhan University. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His area of interest includes perception system of intelligent vehicle.

Martin Lauer received the Diploma degree in computer science from Karlsruhe University and the Ph.D. degree in computer science from Osnabrück University in 2004. He was a Post-Doctoral Researcher with Osnabrück University in machine learning and autonomous robots. Since 2008, he has been the Head of the Research Group, Karlsruhe Institute of Technology. His main research interests are in the areas of machine vision, autonomous vehicles, and machine learning.