IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 9, 2019, accepted January 1, 2020, date of publication January 6, 2020, date of current version January 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964100

Deep Ensemble Object Tracking Based on Temporal and Spatial Networks

ZHAOHUA HU[1,2], HUXIN CHEN[1], AND GAOFEI LI[1]

[1]School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China
[2]Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China

Corresponding author: Zhaohua Hu (zhaohua_hu@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61601230, in part by the Natural Science Foundation of Jiangsu Province, China, under Grant BK20141004, and in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

ABSTRACT In recent years, correlation filtering and deep learning have achieved good performance in object tracking. Correlation filtering is an efficient and real-time method because its formula provides a fast solution in the Fourier domain, but it does not benefit from end-to-end training. Although deep learning is an effective method for learning object representations, training deep networks online with one or a few examples is challenging. To address these problems, we propose a deep ensemble object tracking algorithm that fuses temporal and spatial information to improve algorithm precision and robustness. The framework of our algorithm includes four aspects: feature extraction, a baseline network, a branch network and adaptive ensemble learning. Feature extraction extracts the general object representation. The baseline network integrates feature extraction and a correlation filtering algorithm into a convolutional neural network for end-to-end training. The branch network is composed of a temporal network and a spatial network. The temporal and spatial networks capture the object temporal and spatial information and further refine the object position. Our algorithm only needs an initial frame to train all networks. Adaptive ensemble learning compensates for the object information deficiency and improves tracking accuracy. Many experiments on tracking benchmark datasets demonstrate that our algorithm performs favourably compared with state-of-the-art tracking algorithms.

INDEX TERMS Object tracking, feature extraction, baseline network, temporal and spatial networks, adaptive ensemble learning.

I. INTRODUCTION

Visual object tracking is a fundamental problem in the computer vision field. Visual object tracking can be widely used in many practical systems, such as unmanned aerial vehicle (UAV) [1], video surveillance [2], and human-computer interaction [3]. The foundation of this problem is developing a robust appearance model with extremely limited training data (usually the bounding box in the first frame). Although visual object tracking technology has made considerable progress in the past decades, illumination variation, motion blur, in-plane rotation, low-resolution, and out-of-view are problems that must still be addressed.

In the last three years, deep learning has achieved breakthrough results in object tracking technology. For visual recognition tasks, deep learning models require numerous labelled training samples. However, in visual tracking problems, because the only labelled sample is the object annotated in the first frame, it is infeasible to directly apply a deep learning model to an online tracking algorithm. Previous deep learning-based tracking algorithms required a large number of labelled videos to learn general feature representations by offline training. For example, the multi-domain network (MDNet) tracker [4] uses video sequences from similar tracking benchmarks to pre-train a deep model and uses object benchmark sequences to fine-tune the learned model online. This method is not only prone to overfitting but also consumes too much time for pre-training. A convolutional neural network (CNN) has also been used as an online

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Zhang.

classifier in visual recognition tasks, and the last convolutional layer output is usually used to represent the object. Because features from the last layer have higher semantic information, it is easy to infer the category of the object. Using features from the last layer is effective for visual recognition tasks. However, for visual tracking problems, it is not sufficient to use features from the last layer because the object also needs to be accurately located. These limitations raise two problems. The first problem is how to eliminate offline training in deep learning-based object tracking algorithms, and the second problem lies in how to use the rich feature hierarchies of CNNs to represent the object rather than using features from the last layer.

Recently, correlation filtering methods have attracted considerable attention in the object tracking domain mainly because of the following two characteristics. First, correlation filtering algorithms are efficient approaches that learn to distinguish the object from the searching patches by solving a ridge regression problem. According to the convolution theorem (correlation version), correlation in the spatial domain corresponds to product in the Fourier domain, which allows the ridge regression problem to be solved by the straightforward element-wise operations and the Fast Fourier Transform (FFT) [5], [6]. This property greatly reduces computational complexity and improves the computational speed, so correlation filtering algorithms satisfy real-time requirements. Second, correlation filtering algorithms regress all the circular-shifted versions of the input features to a Gaussian function, so it is always possible to generate dense response scores at the search locations. Correlation filtering-based tracking algorithms have achieved good results on recent tracking benchmark datasets using CNN features, but the existing correlation filtering algorithms have two limitations. First, learning correlation filters and feature extraction are independent of each other; that is, a model is not trained end-to-end. If the extracted features produce errors, they will affect the subsequent task (learning correlation filters) and result in cumulative errors. Therefore, the system trained in this way encounters difficulty achieving the optimal performance. Second, most correlation filtering algorithms are updated in a simple way, basically using linear interpolation to update the learned filters to achieve the model adaptation effect. This method is only an empirical operation; once the noise is updated, it leads to object drift. Two problems ensue with these limitations. The first problem is how to develop an end-to-end training model. The end-to-end approach can avoid cumulative errors caused by the multi-task model and reduce the complexity of the project such that one network solves all tasks. The second problem lies in how to more effectively update the model rather than using empirical operations such as linear interpolation.

To solve the above problems, this paper proposes a deep ensemble object tracking algorithm based on temporal and spatial networks (TSDET). We construct a baseline network for integrating feature extraction and correlation filtering algorithms into a CNN. Feature extraction is carried out by a pre-trained CNN model. Convolution operations in spatial is similar to the dot product between the circular-shifted versions of the input and the correlation filter, so correlation filtering can be redefined as a layer of the CNN to directly generate response mapping as a spatial correlation between consecutive frames. This paper employs a historical sample to learn a temporal network, which can capture the object temporal information. To further explore object spatial information, this paper also constructs a spatial network based on a baseline network to refine the object location. Finally, the temporal network and the spatial network are formed into a new branch network. The new branch network is completely differentiable, which allows the gradient descent algorithm to update convolutional filters. In addition, the search patches generated by the current frame and previous frame are fed into the branch network to construct a weak tracker, and then all weak trackers are combined into a stronger tracker by an adaptive ensemble learning algorithm. This ensemble learning further improves accuracy.

In summary, the main contributions of this paper are as follows:

1) We redefine the correlation filtering as a CNN layer, which ensures that our algorithm is trained end-to-end. We also eliminate offline training to avoid overfitting.
2) We propose a new branch network for the visual tracking problem that combines the object temporal and spatial information to produce high-performance tracking results.
3) We apply adaptive ensemble learning to combine weak trackers from numerous convolutional layers into a strong tracker, which effectively compensates for lacking object information.
4) We develop a new update method to connect adaptive ensemble learning with deep network updating, which improves the precision and robustness of our algorithm.

Based on the above contributions, this paper carries out extensive experiments on four large benchmark datasets, OTB-2013 [31], OTB-2015 [38], VOT2016 [39] and VOT2017 [40]. We prove that the proposed tracking algorithm has superior precision and robustness compared with the existing advanced algorithms.

II. RELATED WORK

In this section, we provide a brief review of correlation filtering-based tracking methods, deep learning-based tracking methods, ensemble learning-based tracking methods and spatial-temporal model-based tracking methods.

A. CORRELATION FILTERING-BASED TRACKING METHODS

Correlation filtering-based tracking methods have attracted considerable attention because of their high computational efficiency. Bolme *et al.* [5] develop the method of minimum output square error (MOSSE) to learn the filters and use intensity features to represent the object. To improve the tracking precision, subsequent researchers optimize the

MOSSE method. Henriques *et al*. [6] propose a kernelized correlation filter (KCF) by introducing the kernel space and employing ridge regression. Additionally, a method that fuses multi-channel features into the correlation filtering is proposed, and the object is represented by the histogram of oriented gradient (HOG) feature. Danelljan *et al*. [7] propose a scale estimation method (DSST) to deal with large variation in the object size. Danelljan *et al*. [8] learn a spatially regularized correlation filter (SRDCF) that overcomes the boundary effect problem caused by circular-shifted samples. Bertinetto *et al*. [9] propose a feature fusion tracking method (Staple), which fuses the HOG feature and the colour histogram feature to further improve the precision. Danelljan *et al*. [10] develop a continuous convolution operator (CCOT), which converts discrete location estimation into continuous location estimation in the time domain. In addition, CCOT combines various hand-crafted features with CNN features to obtain high precision. On the basis of CCOT, Danelljan *et al*. [11] propose an efficient convolution operator (ECO). The method removes redundancy from training samples, filter coefficients and the template update, which sufficiently restrains the redundancy of CNN features and improves the algorithm speed. In the existing correlation filtering-based methods, the tasks of learning the correlation filter and the feature extraction are independent of each other. It indicates that the correlation filtering-based methods cannot benefit from end-to-end training. Different from these methods, the correlation filtering is redefined as a CNN layer in our work, which ensures that our algorithm can be trained end-to-end. In addition, unlike MOSSE and KCF that use linear interpolation to update the learned filter, we develop a new update method which can reduce the error caused by noise and make the model more stable.

B. DEEP LEARNING-BASED TRACKING METHODS

Deep learning-based tracking methods have achieved great success in the object tracking domain by virtue of their excellent feature modelling ability. Wang *et al*. [12] propose a deep learning tracking algorithm (DLT), which first applies offline pre-training and online fine-tuning in the object tracking domain. Because of insufficient training data, the autoencoder network obtains a poor representation ability and the algorithm has low precision. Subsequently, Wang *et al*. [13] optimize the DLT algorithm and propose structured output deep learning tracking (SO-DLT), which transforms the fully connected network into a convolutional neural network, and first applies the CNN model in object tracking. Hong *et al*. [14] learn discriminative saliency map with CNN (CNN-SVM), which combines a CNN model with conventional tracking techniques. Wang *et al*. [15] develop a tracking algorithm with fully convolutional networks (FCNT), which uses features from different layers to effectively restrain the tracker drift. Nam *et al*. [4] propose a multi-domain network (MDNet) to solve the problem of insufficient training data. The MDNet method learns the general feature representation of the object by offline training and fine-tunes the network during online tracking to adapt to the new object changes. Held *et al*. [16] propose a tracking algorithm with deep regression networks (GOTURN), which speed up the algorithm by removing the template update. In addition to the CNN model, object tracking also employs other deep models, such as the Siamese network [17]–[20] and recurrent neural network (RNN) [21], [22]. For example, Bertinetto *et al*. [17] develop fully convolutional Siamese networks (Siamese-FC) to learn a similarity metric offline. Fan *et al*. [22] optimize the MDNet algorithm and propose a structure-aware network (SANet), which captures context information by an undirected recurrent neural network, so that the network can pay more attention to the useful part of tracking. In contrast to the existing deep learning-based methods, we extract CNN features from numerous convolutional layers, which is beneficial to mitigate the effects of dramatic changes in appearance. Meanwhile, we apply temporal and spatial networks to capture the object temporal and spatial information, which helps our algorithm to further refine the object location.

C. ENSEMBLE LEARNING-BASED TRACKING METHODS

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone [23]. Ensemble learning utilizes multiple experts for visual tracking. Most ensemble tracking methods [24] are based on hand-crafted features. Wang *et al*. [25] employ a conditional particle filter to infer the object location and the reliability of each member tracker. Nam *et al*. [41] develop a tree-structured convolutional neural network tracking algorithm (TCNN), which improves the fully connected layer structure into a tree structure and uses multiple fully connected layers with high confidence to determine the object position. Ma *et al*. [26] propose a tracking algorithm with hierarchical convolutional features (HCFT), which combines the rich feature hierarchies of CNNs with the correlation filtering algorithm. It is the first time that CNN features are applied to ensemble learning, and accurate tracking results are obtained. In contrast to these works, we believe that visual tracking is a decision-theoretic learning problem based on multi-expert trackers. In other words, during online tracking, each expert makes a decision, and the weighted decision of all experts determines the final object position. Our model is largely inspired by the HCFT algorithm. We also employ rich feature hierarchies of CNNs to improve tracking accuracy and robustness. However, different from the fixed weights method in HCFT, in this paper, we adopt the weights in an adaptive way and combine our model with the CNN.

D. SPATIAL-TEMPORAL MODEL-BASED TRACKING METHODS

Recently, increasingly more scholars who study object tracking have introduced the spatial and temporal framework into the tracking model because of its excellent performance. For instance, Zhang *et al*. [42] propose a spatiotemporal context tracking algorithm (STC), which is based on a

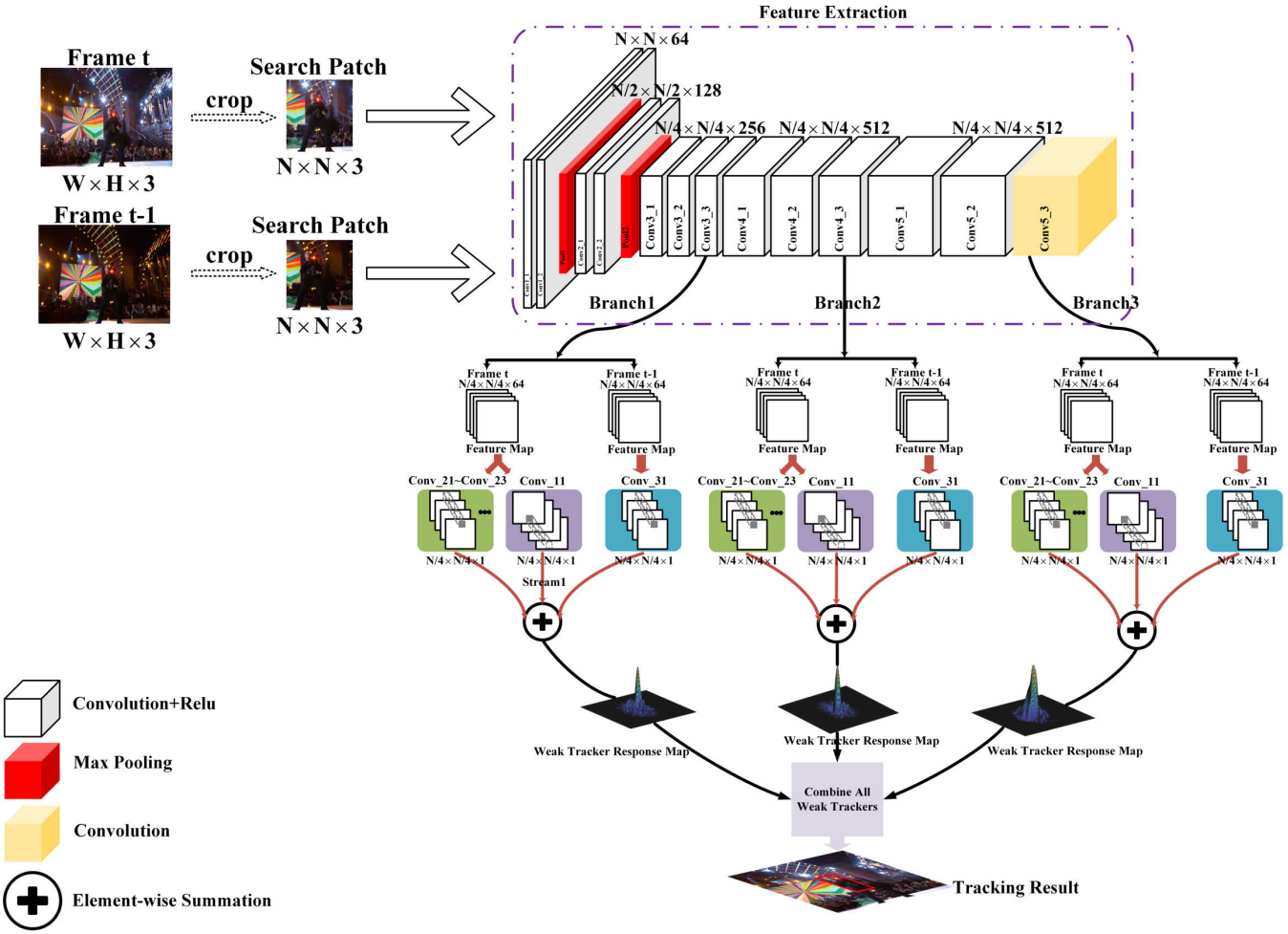


FIGURE 1. The proposed tracking approach pipeline. We set the size N of search patch to five times the maximum value of object width and height. The pre-trained VGGNet [27] is used to extract CNN features from search patches of the current frame and the previous frame. These features from numerous layers are fed into designed convolutional layers (including Conv_11 is detailed in Section III-B, Conv_21–Conv_23 and Conv_31 are delineated in Section III-C) to generate response maps, from which a weak tracker is constructed with moderate performance. Finally, all weak trackers are combined into a strong tracker by adaptive ensemble learning to predict the object location.

Bayesian framework to model the spatiotemporal relationship between the object and its local background. Moreover, the spatiotemporal model learning and the object detection are implemented by FFT. Li *et al.* [43] develop a tracking algorithm with learning spatial-temporal regularization (STRCF), which introduces spatial and temporal regularization into the original KCF formula to handle the boundary effect without loss of efficiency. Liu *et al.* [44] propose a multi-scale spatiotemporal feature tracking algorithm (MSST-ResNet), which combines the spatial feature and temporal feature into the residual units with the multi-scale feature to directly learn the object representation. Zhu *et al.* [45] construct a deep spatiotemporal feature learning algorithm based on the correlation filtering framework (STResNet_CF). To utilize the spatial context information and the temporal relationship of successive frames, the algorithm learns the spatial feature and temporal feature separately. Different from the above algorithms, the spatial and temporal features of our algorithm are directly applied in the CNN, rather than in correlation filtering. We extend the VGGNet to the spatial and temporal networks to extract appearance features of the object. Specifically, unlike MSST-ResNet and STResNet_CF, our backbone network is based on VGGNet [27] rather than ResNet [46]. In addition, the time-consuming offline training is used in their algorithms, but our algorithm only needs an initial frame for training, which improves the efficiency and demonstrates that the proposed method has good generalization performance.

III. PROPOSED ALGORITHM

The pipeline of the deep ensemble object tracking algorithm based on temporal and spatial networks proposed in this paper is shown in Fig. 1. The algorithm consists of four main parts: extracting CNN features (Section III-A), building a baseline network (Section III-B), constructing a branch network with temporal and spatial networks (Section III-C) and combining all weak trackers (Section III-D). In the following,

we first briefly introduce the CNN features and the baseline network, then we elaborate the branch network with temporal and spatial networks, which are the core components of our algorithm, finally we illustrate the adaptive ensemble learning.

A. CNN FEATURES

In recent years, a large number of CNN models have been used for large-scale image classification and visual recognition tasks, such as LeNet [28], AlexNet [29], GoogLeNet [30] and VGGNet [27]. This paper employs VGGNet to extract features. Compared with AlexNet, VGGNet has a deeper network structure. VGGNet successfully constructs a 16–19 layers CNN, and the network has strong expansibility and excellent generalization ability for migrating to track images. In addition, VGGNet is trained using 1.3 million images from the ImageNet dataset, which achieves robust results in image classification tasks.

Image classification and visual recognition tasks require the extracted features to have more semantic information. In contrast to these tasks, object tracking requires not only semantic information but also extracted features with precise localization capabilities. Therefore, as analyzed in [26], this paper also visualizes the up-sampled feature maps of Conv3_3, Conv4_3, and Conv5_3 layers using the VGGNet to illustrate the information represented by each layer feature. We use the MountainBike sequence in the benchmark dataset [31]. As shown in Fig. 2, the red bounding boxes indicate the tracking results. The profile of the object and the background can be seen in Fig. 2(b) and (c). We notice that features on the third and fourth layers have higher spatial resolution and are useful for precisely locating the object. Additionally, it is difficult to see the details of the object in Fig. 2(d). As shown in frames 135 and 228 of Fig. 2(d), despite the dramatic object changes, features on the fifth layer can effectively distinguish the object from the background. In other words, the shallow CNN features provide object spatial information, whereas the deep CNN features contain more semantic information to deal with intense object changes and prevent tracker drift. In summary, VGGNet achieves a more detailed description of the object by extracting features from different layers. Since the extracted feature channels from different layers are too large to fit our network effectively, we need to adopt deep feature selection. Recently, Nalepa *et al*. [57] introduces a method for automatic deep feature selection, which combines a genetic algorithm with deep learning to reduce the number of feature channels. We think this method is novel, but it adds a higher level of complexity, similar to that encountered in the traditional genetic algorithm, which is inconvenient when we test the performance of our algorithm. Therefore, we reduce the extracted feature channels by principal component analysis (PCA) to accelerate the network learning. PCA is a commonly used dimensionality reduction algorithm, which works well in practice. It is fast and simple to implement.

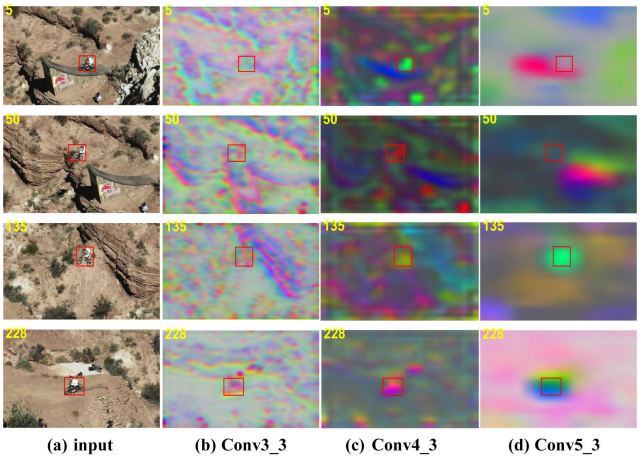


FIGURE 2. Visualization of convolutional layers. (a) Four frames from the MountainBike sequence. (b)–(d) Features are extracted on convolutional layers Conv3_3, Conv4_3, and Conv5_3 through the VGG-Net [27].

B. BASELINE NETWORK

Fig. 3 shows the architecture of our baseline network, which corresponds to a stream in Fig. 1. First, we review the correlation filtering fundamentals and change it to a network structure. The principle of correlation filters is as follows. In the first frame, a general filter template is learned. In the next frame, a response map is generated using the filter template, and the object position is predicted by searching the maximum value in the response map. In fact, learning a correlation filter Ω by minimizing the following function:

$$\widehat{\Omega} = \arg\min_{\Omega} \|\Omega * X - Y\|^2, \tag{1}$$

where $*$ denotes the convolution operation, X is the input sample, and Y denotes the corresponding Gaussian label. To prevent overfitting, l_2 regularization is introduced into the formula, and then (1) is modified to the following form:

$$\widehat{\Omega} = \arg\min_{\Omega} \|\Omega * X - Y\|^2 + \lambda\|\Omega\|^2, \tag{2}$$

where λ is the regularization parameter. To solve the ridge regression problem (2), a method is proposed in [6] to generate the input samples by cyclic shifting from a search patch, and then the closed-form solution of (2) is obtained in the Fourier domain. This method is efficient, but the circular-shifted samples can lead to the boundary effect. Therefore, we try to explain (2) from another point of view.

We redefine the learning process of the above correlation filtering as a CNN cost function minimization problem. The general form of the cost function is as follows:

$$J(\Theta) = \frac{1}{M}\sum_{i=1}^{M} L\left(Y^{(i)}, F\left(X^{(i)}; \Theta\right)\right) + \lambda P(\Theta), \tag{3}$$

where Θ is the convolution filter, M is the number of samples, $X^{(i)}$ represents the i-th input sample, $Y^{(i)}$ denotes the ground-truth label corresponding to the i-th sample, $L(\cdot)$ represents the loss function of the i-th sample, $F(\cdot)$ denotes the network output of the i-th sample, which is the predicted

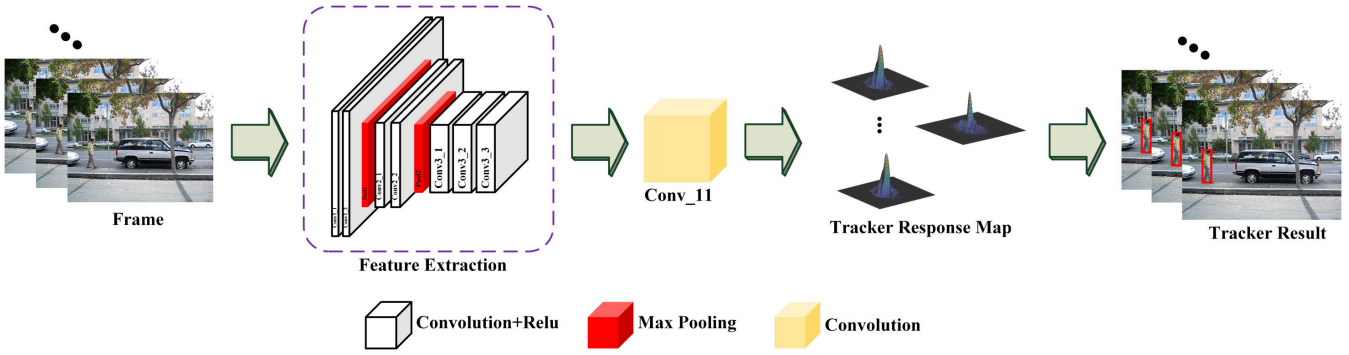


FIGURE 3. The architecture of our baseline network.

value, λ is the regularization parameter, and $P(\cdot)$ is the regularization function. Let $M = 1$, take the l_2 loss function as $L(\cdot)$, set the l_2 norm as the regularization function, and (3) can be rewritten as follows:

$$J(\Theta) = \|F(X;\Theta) - Y\|^2 + \lambda\|\Theta\|^2. \tag{4}$$

When the input sample X passes through a convolutional layer, its network output is $F(X;\Theta) = \Theta * X$ ($*$ represents the convolution operation). Comparing (4) and (2), we find that the correlation filter Ω is equivalent to the convolution filter Θ in the network, and the objective function in the correlation filtering algorithm is also equivalent to the cost function in the network. Therefore, we redefine the correlation filtering as one convolution layer Conv_11. As shown in Fig. 3, the tracking model and feature extraction are integrated into the CNN. We adopt the first frame and the Gaussian label as a training pair, which is fed into our baseline network for training. In addition, the gradient descent algorithm and backpropagation can be used to obtain the weights instead of the closed-form solution.

C. BRANCH NETWORK WITH TEMPORAL AND SPATIAL NETWORKS

We discussed the baseline network in Section III-B. Ideally, the baseline network output should be the same as the ground truth. In reality, it is difficult to achieve this using feature extraction and one convolutional layer. To solve this problem, the traditional solution is to deepen the network, but in the tracking domain, deeper networks affect real-time tracking. Therefore, we propose a shallow branch network with temporal and spatial networks to capture the object temporal and spatial information, which effectively reduces the difference between the baseline network output and the ground truth.

Fig. 4 shows the architecture of our branch network. In Fig. 1, the output of the feature extraction network has three branches, and Fig. 4 corresponds to one of the branches. As shown in Fig. 4, the proposed spatial network consists of the baseline network and three convolutional layers: Conv_21, Conv_22 and Conv_23. The first two convolutional layers, Conv_21 and Conv_22, have ReLU non-linearities, and the pooling layer is not added in all three convolutional layers because there is no need to compress the feature map and we want to retain more spatial resolution on each convolutional layer. Additionally, the filter sizes of the three convolutional layers are the same, and all adopt a 1×1 filter, but the number of channels is different. Using a 1×1 filter is a common technique in the field of deep learning. Its function can be roughly divided into the following two aspects: realizing cross-channel interaction and information integration; implementing feature channels dimensionality ascent and reduction.

In the spatial network designed in this paper, we feed the current frame into the feature extraction network. Since spatial information is attenuated as a network becomes deeper [14], to solve this problem, we design three convolutional layers. The Conv_21 layer is used to integrate spatial information, which is from the feature extraction network output. To increase the nonlinearity of the network and enable the network to express more complex features, the Conv_22 layer is adopted. The Conv_23 layer is used to reduce the feature channels dimensionality. We retrieve the output map after the Conv_23 layer and fuse it into the baseline network output by element-wise summation. According to the network structure in Fig. 4, our spatial network output is formulated as follows:

$$\Phi_S^{[1]}(X_t) = relu\left(W_S^{[1]} * X_t + B_S^{[1]}\right), \tag{5}$$

$$\Phi_S^{[2]}(X_t) = relu\left(W_S^{[2]} * \Phi_S^{[1]}(X_t) + B_S^{[2]}\right), \tag{6}$$

$$\Phi_S^{[3]}(X_t) = W_S^{[3]} * \Phi_S^{[2]}(X_t) + B_S^{[3]}, \tag{7}$$

$$F_{spatial}(X_t) = \Phi_S^{[3]}(X_t) + F_{base}(X_t), \tag{8}$$

where $*$ denotes the convolution operation, X_t represents the feature map of the current frame, $W_S^{[1]}$, $W_S^{[2]}$ and $W_S^{[3]}$ represent the filters for the Conv_21, Conv_22 and Conv_23 layers, respectively, $B_S^{[1]}$, $B_S^{[2]}$ and $B_S^{[3]}$ denote the biases for the Conv_21, Conv_22 and Conv_23 layers, respectively, and $\Phi_S^{[1]}(X_t)$, $\Phi_S^{[2]}(X_t)$ and $\Phi_S^{[3]}(X_t)$ denote the outputs of the Conv_21, Conv_22 and Conv_23 layers, respectively. $F_{base}(X_t) = W_{base} * X_t + B_{base}$ is the baseline network output where W_{base} and B_{base} denote the filter and bias, respectively, for the Conv_11 layer.

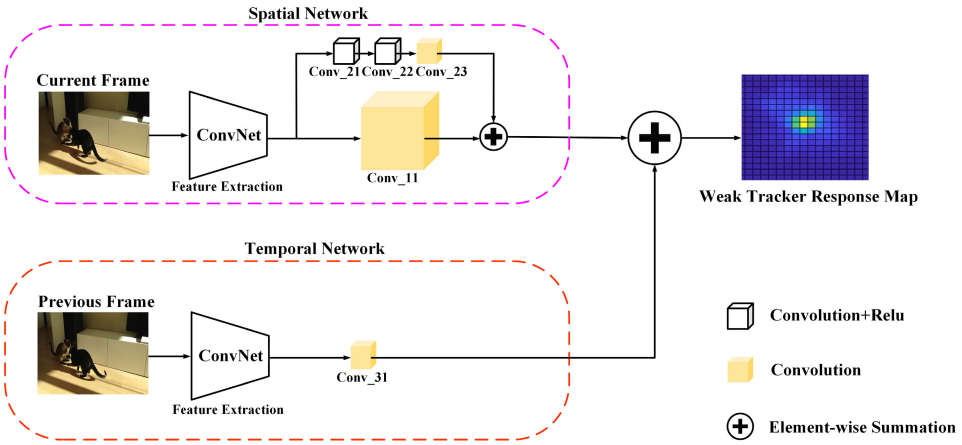


FIGURE 4. The architecture of our branch network with temporal and spatial networks.

In addition, the spatial network can capture only the current frame spatial information. When the object changes dramatically, the model learned by only the spatial information is unstable. Therefore, we also design a temporal network to capture the previous frame information. We refer to the previous frame information as temporal information. As shown in Fig. 4, the temporal network is composed of feature extraction and one convolutional layer, Conv_31, which uses a 1×1 filter. The output expression for the temporal network is as follows:

$$F_{temporal}\left(X_{t-1}\right) = W_T * X_{t-1} + B_T, \tag{9}$$

where $*$ denotes the convolution operation, X_{t-1} represents the feature map of the previous frame, W_T is the filter for the Conv_31 layer, and B_T is the bias for the Conv_31 layer.

To integrate the captured temporal information into the baseline network, as shown in Fig. 4, we fuse the temporal network output directly with the spatial network output by element-wise summation. At the same time, the channel dimension of each layer network output needs to be consistent when fusion is adopted. Finally, our branch network output can be formulated as follows:

$$F\left(X_t\right) = F_{spatial}\left(X_t\right) + F_{temporal}\left(X_{t-1}\right). \tag{10}$$

Generally, when the object changes slightly, the difference between the baseline network and the ground truth is small, so the temporal and spatial networks have little effect. However, when the object moves violently, such as fast motion, the response from the baseline network is limited, which leads to an inability to distinguish the object from the background. Combining the temporal and spatial networks alleviates this limitation, which helps reduce the noise response in final output and makes the final response closer to the ground truth. Therefore, the response of the combined temporal and spatial networks is more robust for violent object changes.

D. ADAPTIVE ENSEMBLE LEARNING

We illustrate an adaptive ensemble learning algorithm to solve the multi-expert decision-theoretic online learning problem. In each frame, a final decision is made by the weighted of all experts. In the next frame, the weight of each expert is updated to reflect the decision loss of each expert. In Section III-C, we described in detail the branch network with temporal and spatial networks, which can be regarded as a weak tracker model. In visual tracking problems, a weak tracker is considered an expert. In the t-th frame, the response of the l-th expert is calculated by:

$$F_t^l = F_{spatial}^l\left(X_t^k\right) + F_{temporal}^l\left(X_{t-1}^k\right), \quad k = 3, 4, 5, \tag{11}$$

where $k = 3, 4, 5$ denotes the third, fourth and fifth layers in VGGNet, F_t^l is a Gaussian function. The final response is weighted by multiple experts:

$$E_t^l = \sum\nolimits_{l=1}^{L} \theta_t^l \cdot F_t^l, \tag{12}$$

where θ_t^l is the weight of the l-th expert and $\sum_{l=1}^{L} \theta_t^l = 1$. The predicted object position in the t-th frame is computed by:

$$\left(x_t^p, y_t^p\right) = \arg\max_{x_t, y_t} E_t^l\left(x_t, y_t\right), \tag{13}$$

where $E_t^l\left(x_t, y_t\right)$ denotes the element at position $\left(x_t, y_t\right)$ of the final response E_t^l. Once the final object position is available, the decision loss of each expert can be calculated. The calculation formula is as follows:

$$C_t^l = \max\left(F_t^l\right) - F_t^l\left(x_t^p, y_t^p\right), \tag{14}$$

where $F_t^l\left(x_t^p, y_t^p\right)$ denotes the element at position $\left(x_t^p, y_t^p\right)$ of expert's response F_t^l. To obtain a new weight distribution, we introduce a regret measure [32] commonly used in decision-theoretic. The regret measure is defined as the regret (of the learner) to an action and is the difference between

the learner's cumulative loss and the cumulative loss of that action [32], which can be expressed as follows:

$$m_t^l = \sum_{l=1}^{L} \theta_t^l C_t^l - C_t^l = \overline{C_t^l} - C_t^l. \tag{15}$$

In the object tracking field, because the appearance of the object usually changes at an irregular speed, it is necessary to use the previous regret (historical cumulative regret M_{t-1}^l). Additionally, since each expert represents different aspects of the object, the historical regret with different weights needs to be allocated. According to this idea, a new objective function is proposed. Learning the new weight of each expert is also transformed into the new objective function minimization problem. Because the appearance of the object changes slightly over a short time, a Gaussian distribution with mean μ_t^l and standard variance s_t^l is used to model the stability of each expert in Δt time before finding the new objective function. The stability of the l-th expert is calculated by the following formula:

$$\eta_t^l = \frac{\left|C_t^l - \mu_t^l\right|}{s_t^l}, \tag{16}$$

where the mean $\mu_t^l = \frac{1}{\Delta t} \sum_{\tau=t-\Delta t+1}^{t} C_\tau^l$, and the standard variance $s_t^l = \sqrt{\frac{1}{\Delta t-1} \sum_{\tau=t-\Delta t+1}^{t} \left(C_\tau^l - \mu_t^l\right)^2}$. When the value of η_t^l is small, it indicates that the expert tends to be stable so we assign a large weight to the current frame regret. In contrast, if the value of η_t^l is relatively large, it indicates that the expert's performance is poor, and it is necessary to assign a large weight to the historical regret. Based on this principle, we obtain the following new objective function (cumulative regret):

$$M_t^l = \beta_t^l m_t^l + \left(1 - \beta_t^l\right) M_{t-1}^l, \tag{17}$$

$$\beta_t^l = \min\left(H, \exp\left(-\zeta \eta_t^l\right)\right), \tag{18}$$

where ζ is a scale factor that controls the shape of the exponential function, H represents the maximum weight of the current frame regret to avoid no historical regret.

In [32], a potential function $\phi\left(M_t^l, \sigma_t\right)$ is computed by the quadratic regret, which can be written as:

$$\phi\left(M_t^l, \sigma_t\right) = \exp\left(\frac{\left(\left[M_t^l\right]_+\right)^2}{2\sigma_t}\right), \tag{19}$$

where $\left[M_t^l\right]_+$ denotes $\max\left(0, M_t^l\right)$, and σ_t is a scale parameter, which is computed by solving $\frac{1}{L} \sum_{l=1}^{L} \exp\left(\frac{\left(\left[M_t^l\right]_+\right)^2}{2\sigma_t}\right) = e$. Learning a new weight θ_{t+1}^l by minimizing the cumulative regret (17). The closed-form solution of minimizing (17) is set proportionally to the first derivative of the potential function. The new weight is as follows:

$$\theta_{t+1}^l \propto \frac{\left[M_t^l\right]_+}{\sigma_t} \exp\left(\frac{\left(\left[M_t^l\right]_+\right)^2}{2\sigma_t}\right). \tag{20}$$

IV. TRACKING PROCESS

This section describes our algorithm's detailed object tracking process. We present the tracking process through training, online detection, and model update.

A. TRAINING

The algorithm proposed in this paper does not require time consuming offline training; it needs only an input frame (first frame) with the object position. A Gaussian-shaped label is generated by the object position. Then, we extract a training patch that is centred on the object position and feed it into our framework for feature extraction and response mapping. We regard the training patch and the Gaussian label as training pairs. VGGNet is used for feature extraction. Additionally, the parameters of the Conv_11, Conv_21, Conv_22, Conv_23 and Conv_31 layers are randomly initialized to a zero-mean Gaussian distribution. The temporal network and the spatial network do not need to be trained separately. The training patch of the first frame obtains the feature maps on the third, fourth and fifth layers through VGGNet. The feature maps from these three layers are sent into our designed convolutional layers. We train three models simultaneously until convergence.

B. ONLINE DETECTION

We extract a search patch from the current frame image. We also extract another search patch from the previous frame image. These search patches are centred on the object position predicted by the previous frame and are the same size as the training patch. VGGNet is used to extract the feature maps of the above search patches. We feed the extracted feature maps into the corresponding three trained models to generate response maps. The adaptive ensemble learning algorithm is used to combine all model response maps to obtain the ultimate object position. After we predict the object position, the object size is predicted by the scale estimation algorithm. We extract candidate patches on different scales. These candidates have the same size as the training patch. We send these candidates into the model (feature maps from the third layer) to generate the response maps. Once we have the response maps, we predict the object size by searching for the maximum response value.

C. MODEL UPDATE

We propose a new model update method, which is different from empirical operations such as linear interpolation. As shown in Fig. 5, the new model update adopts a combination of long-term updates and short-term updates. We develop short-term updates to connect adaptive ensemble learning with model updates. Specifically, we compute the model

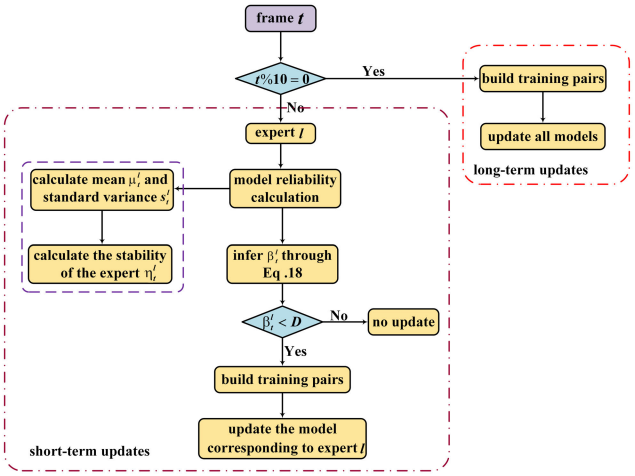


FIGURE 5. The proposed model update method.

TABLE 1. Architecture of the convolutional embedding function.

Layers	Size	Channels	Stride	Activation
Conv1_1	3×3	64	1	ReLU
Conv1_2	3×3	64	1	ReLU
Conv2_1	3×3	128	1	ReLU
Conv2_2	3×3	128	1	ReLU
Conv3_1	3×3	256	1	ReLU
Conv3_2	3×3	256	1	ReLU
Conv3_3	3×3	256	1	ReLU
Conv4_1	3×3	512	1	ReLU
Conv4_2	3×3	512	1	ReLU
Conv4_3	3×3	512	1	ReLU
Conv5_1	3×3	512	1	ReLU
Conv5_2	3×3	512	1	ReLU
Conv5_3	3×3	512	1	–
Conv_11	$fw \times fh$	1	1	–
Conv_21	1×1	64	1	ReLU
Conv_22	1×1	64	1	ReLU
Conv_23	1×1	1	1	–
Conv_31	1×1	1	1	–

reliability, which is the same as the stability of computational experts in adaptive ensemble learning. A smaller η_t^l means that this expert tends to be stable, and the model corresponding to this expert does not need to be updated. In contrast, a larger η_t^l shows that the β_t^l is small, which can be inferred from (18). When the β_t^l is smaller than a threshold D, it indicates that this expert's performance is poor, and we need to update the model corresponding to this expert. We adopt the search patches from online detection and the predicted result as training pairs which are fed into the model for short-term updates. In addition, we also append long-term updates; that is, the three models are all updated every 10 frames. The training data used for long-term updates are the search patches and the predicted results that are continuously generated during the online detection. These two kinds of updates make our algorithm more robust.

V. EXPERIMENTS

In this section, we first present the experimental settings, and then analyse the effect of each component in our tracker. Finally, we compare our tracker with state-of-the-art trackers on the benchmark datasets for performance evaluation.

A. EXPERIMENTAL SETTINGS

1) IMPLEMENTATION DETAILS

The size of the training patch obtained from the first frame is five times the maximum value of the object width and height. The feature extraction network comes from the pre-trained VGG-16 [27], which removes the last three pooling layers. We also delete the fully connected layers because we do not need to classification. We extract features from the Conv3_3, Conv4_3, and Conv5_3 layers and then reduce the extracted feature channels to 64 by PCA. We employ the MatConvNet toolbox [33] in our experiments. The hardware platform uses an Intel i7-8700 3.2 GHz CPU, 8 GB of RAM and a NVIDIA GTX 1060 GPU. Based on a large number of experiments, the filter size of the Conv_11 layer can be calculated as $fw = 2 \times \lceil w/2 \rceil + 1, fh = 2 \times \lceil h/2 \rceil + 1$, where w and h denote the object width and height, $\lceil \cdot \rceil$ is the round up calculation. The reason for this calculation is that we need to make sure that its filter size covers the target object. Table 1 shows the parameters of the main convolutional layers in Fig. 1. In the training phase, the learning rate is set to $5e-8$. In the online detection phase, a period of time in adaptive ensemble learning is set to $\Delta t = 5$, the scale factor in (18) is set to $\zeta = 10$, the maximum weight of the current frame regret in (18) is set to $H = 0.97$, and three scales in the scale estimation algorithm are set to $(1, 0.95, 1.05)$. In the model update phase, the threshold is set to $D = 0.12$, and the learning rate of this phase is set to $2e-9$. All phases use the Adam [34] algorithm for optimization.

2) BENCHMARK DATASETS

To better evaluate our algorithm performance, we employ four large benchmark datasets: OTB-2013 [31], OTB-2015 [38], VOT2016 [39] and VOT2017 [40]. The first two datasets consist of 50 and 100 sequences. These sequences involve 11 video attributes, such as illumination variation (IV), scale variation (SV), object occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out-of-view (OV), background clutter (BC) and low resolution (LR). The last two datasets each contain 60 sequences with six challenging attributes (*i.e.,* occlusion, illumination change, motion change, size change, camera motion, and empty). In the VOT2017, the 10 challenging sequences in the VOT2016 are replaced by new difficult ones.

3) EVALUATION METRICS

For the object tracking benchmarks (OTB) [31], [38], we use one-pass evaluation (OPE) with distance precision (DP) and overlap success (OS) rate metrics. The DP metric refers to the percentage of frames in which the estimated locations are within a given threshold distance of the ground-truth positions [31]. The OS rate metric refers to the overlap

ratio between predicted and ground-truth bounding boxes. Experimental results are reported by precision plots and success plots, which rank trackers in terms of DP score at a threshold of 20 pixels and area under the curve (AUC) score, respectively. For the visual object tracking (VOT) benchmarks [39], [40], we evaluate the performance in terms of accuracy, robustness and expected average overlap (EAO) metrics. The accuracy metric refers to the average overlap rate while tracking successfully, similar to the OS metric in OTB. The robustness metric refers to the number of tracking failures, where a failure is judged to have occurred if the overlap rate is 0. The EAO metric is the expected value of the no-reset overlap for each tracker on a short-term sequence. A good tracker has high accuracy and EAO scores but low robustness.

B. ABLATION STUDIES

To demonstrate the effect of the different components used in our tracker, we perform ablation studies on the OTB-2015 benchmark. Six trackers are shown in this experiment: the tracker with the designed baseline network, which only consists of features from the third layer and one convolutional layer (namely, BaselineNet); based on the baseline network, the tracker adds an adaptive ensemble learning strategy to integrate features from numerous convolutional layers (namely, BaselineNet+ensemble); the trackers with the branch network, which integrates the temporal and spatial networks, whose feature maps are derived from the third, fourth and fifth layers through VGGNet [27] (namely, Conv3_BranchNet, Conv4_BranchNet, Conv5_BranchNet); and our proposed tracker with the branch network and adaptive ensemble learning strategy (namely, Ours). Note that when the adaptive ensemble learning module is removed, the short-term updates are also deleted, so we adopt long-term updates in these trackers.

The comparative results are reported in Fig. 6. As shown in Fig. 6, compared with the BaselineNet tracker, the tracker named 'BaselineNet+ensemble' has a gain of 3.8% in DP scores and 3.6% in the success rate, which indicates that the tracker performance can be improved effectively by adding adaptive ensemble learning. The Conv3_BranchNet tracker surpasses the BaselineNet tracker by 6.4% in DP scores and 5.8% in the success rate. In addition, the Conv4_BranchNet and Conv5_BranchNet trackers obtain better tracking performance in terms of DP scores and OS rate when compared with the BaselineNet tracker, which demonstrates that the tracker obtains obvious improvement through the branch network with two sub-networks (temporal and spatial networks). Compared with the Conv5_BranchNet tracker, the performance of the Conv3_BranchNet and Conv4_BranchNet trackers have an increase in DP scores and OS rate, which can be attributed to the shallow features having higher resolution, to accurately locate the object position. Moreover, because of the applied branch network and the adaptive ensemble learning strategy, our tracker ranks first in Fig. 6 and exceeds the BaselineNet tracker by 8% in DP scores and 6.5% in the success rate. Fig. 7 shows the tracking results of our tracker and comparison trackers on 2 challenging sequences. In each video frame from Fig. 7, our tracker accurately predicts the object position and scale. Finally, we conclude that the branch network with two sub-networks and the adaptive ensemble learning strategy are effective for tracking.

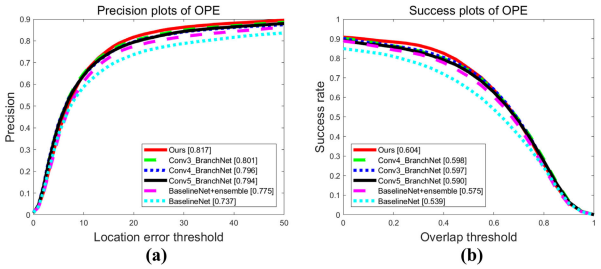


FIGURE 6. Ablative experiments on the OTB-2015 benchmark. (a) Precision plots of OPE. (b) Success plots of OPE.

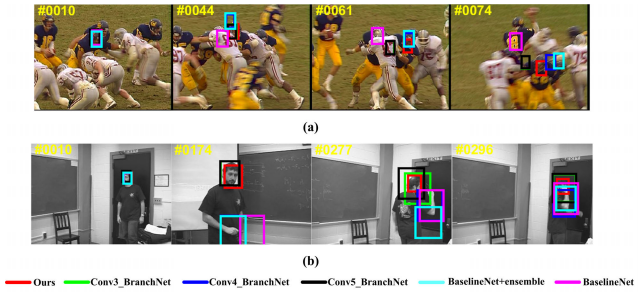


FIGURE 7. Tracking results of our tracker and comparison trackers on 2 challenging sequences. (a) Football1 and (b) Freeman1.

C. COMPARISON WITH STATE-OF-THE-ART TRACKERS

We compare our tracker with state-of-the-art trackers on four benchmarks including OTB-2013, OTB-2015, VOT2016 and VOT2017. Experimental results and analyses are discussed in the following.

1) EXPERIMENTS ON OTB

In this experiment, we select 8 state-of-the-art trackers for comparisons: TRACA [35], ACFN [36], CFNet [19], SiamFC [17], SCT [37], Staple [9], SRDCF [8], CNN-SVM [14]. TRACA combines deep learning and correlation filtering; ACFN is a deep learning method based on an attention structure; CFNet and SiamFC are deep learning methods based on a Siamese network; SCT is a correlation filtering method based on an attention structure; Staple and SRDCF are correlation filtering methods; and CNN-SVM is a deep learning method.

a: OTB-2013 RESULTS

Fig. 8 shows the OPE evaluation results on the OTB-2013 benchmark using the DP scores and OS rate. Statistics between different trackers are summarized in Table 2. As shown in Table 2, the tracker that ranks first on the

TABLE 2. Comparisons between our tracker and state-of-the-art trackers on the OTB benchmarks. Distance precision (DP) and overlap success (OS) rate are reported. The first and second best scores are highlighted by bold and underline.

	Tracker	Ours	TRACA [35]	ACFN [36]	CFNet [19]	SiamFC [17]	SCT [37]	Staple [9]	SRDCF [8]	CNN-SVM [14]
OTB-2013	OS (%)	63.6	**65.2**	60.7	61.0	60.7	59.5	59.3	62.6	59.7
	DP (%)	87.2	**89.8**	86.0	82.2	80.9	83.9	78.2	83.8	85.2
OTB-2015	OS (%)	**60.4**	60.0	57.1	58.9	58.2	53.1	57.9	59.8	55.4
	DP (%)	**81.7**	81.2	79.5	77.7	77.1	76.4	78.4	78.9	81.4

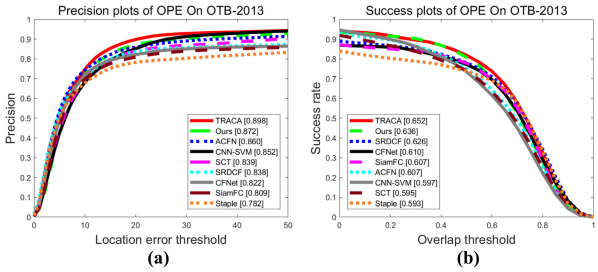


FIGURE 8. Evaluation results on the OTB-2013 benchmark. (a) Precision plots of OPE. (b) Success plots of OPE.

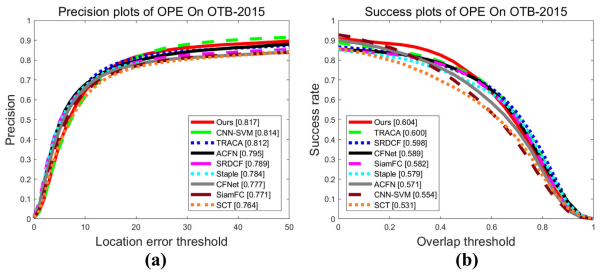


FIGURE 9. Evaluation results on the OTB-2015 benchmark. (a) Precision plots of OPE. (b)Success plots of OPE.

OTB-2013 is TRACA, which achieves a DP of 89.8%, and a success rate of 65.2%. Our tracker takes the second place and obtains a DP of 87.2% and a success rate of 63.6%. Although the performance of our proposed tracker is inferior to TRACA, our tracker performs better than other trackers used for comparison. For example, our tracker surpasses ACFN by 1.2% in DP scores and 2.9% in the success rate. As can be seen from Table 2, our tracker is superior to Siamese network-based trackers SiamFC and CFNet, as well as the trackers based on correlation filtering Staple and SRDCF according to DP scores and OS rate. Compared with the CNN-SVM, our tracker has a gain of 2.0% in DP scores and 3.9% in the success rate. In addition, our tracker outperforms SCT by 3.3% in DP scores and 4.1% in the success rate. Experimental results indicate that our tracker achieves good performance among these state-of-the-art trackers.

b: OTB-2015 RESULTS

The OPE evaluation results on the OTB-2015 benchmark are reported in Fig. 9. Fig. 9 illustrates that our tracker performs favourably against the state-of-the-art trackers in terms of DP scores and OS rate. Meanwhile, more details in Fig. 9 are summarized in Table 2. As shown in Table 2, our tracker ranks first on the OTB-2015 and achieves a DP of 81.7% and a success rate of 60.4%. Moreover, although our tracker performs inferior to TRACA on the OTB-2013, it has a better performance on the OTB-2015 compared with TRACA, which can be seen in Table 2. Since the OTB-2015 benchmark involves more fast motion and motion blur video sequences, TRACA is not as good at dealing with these sequences as our tracker. Compared with CNN-SVM that ranks second among nine trackers in DP scores, our tracker has a gain of 5.0% in the success rate. Our tracker surpasses ACFN by 3.3% and SCT by 7.3% in the success rate. In addition, our tracker performs better than the correlation filtering-based trackers SRDCF and Staple. Similarly, the highest DP scores and success rate of our tracker demonstrate the superiority over CFNet and SiamFC, which are based on the Siamese network. In summary, our tracker obtains the best performance among these trackers used for comparison.

To evaluate the proposed tracker robustness in various scenarios, we further analyse the tracker performance under different video attributes. Fig. 10 shows evaluation plots that contain six video attributes: FM, MB, IV, DEF, OPR, and OCC. As shown in Fig. 10, our tracker is effective in dealing with FM, MB, IV and DEF, which is attributed to the fact that our tracker considers feature representation from numerous convolutional layers, builds the branch network with two sub-networks (temporal and spatial networks) and employs a new update strategy. Our tracker performs similarly to TRACA when handling OPR sequences. Because TRACA utilizes data augmentation, our tracker is inferior to TRACA in the DP metric. In addition, our tracker does not perform as well as TRACA in the case of OCC, which can be attributed to the lack of a module in our tracker to specifically deal with OCC. TRACA adopts a re-detection model to handle OCC, so the re-detection model will be considered in our future work.

2) EXPERIMENTS ON VOT

a: VOT2016 RESULTS

We evaluate our tracker on the VOT2016 benchmark by comparing it with 14 released state-of-the-art trackers, including VITAL [47], ECO-HC [11], Staple_p [39], SiamRN [39], DNT [48], DeepSRDCF [49], MDNet_N [4], RFD_CF2 [50], SiamAN [39], deepMKCF [51], HCFT [26], KCF [6],

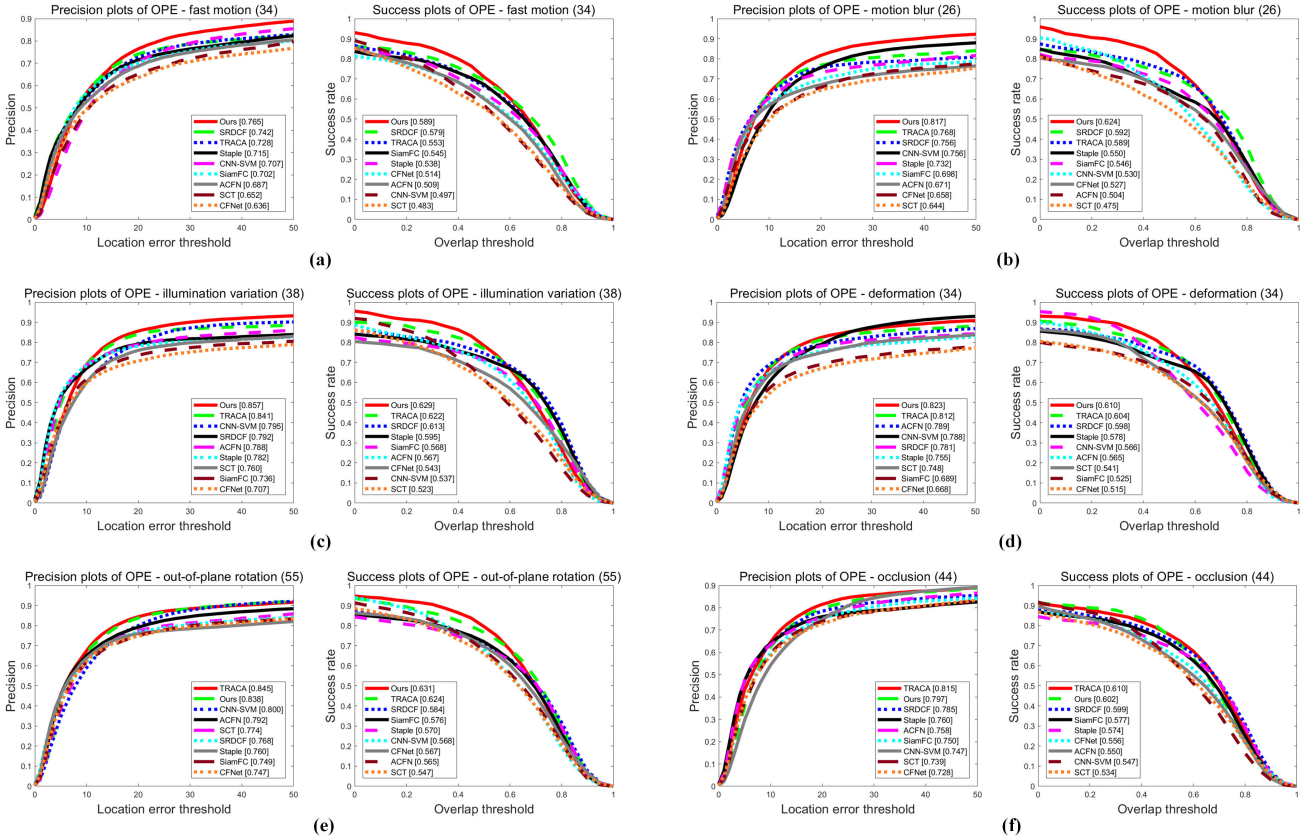


FIGURE 10. **Evaluation results on different video attributes, including (a) fast motion, (b) motion blur, (c) illumination variation, (d) deformation, (e) out-of-plane rotation and (f) occlusion.**

SAMF [52], and STC [42]. Note that in this experiment, both SiamRN and SiamAN refer to the SiamFC algorithm. The difference is that SiamAN uses the AlexNet architecture for the embedding function, while SiamRN employs the ResNet architecture instead of AlexNet as the backbone network. In addition, MDNet_N is a variation of MDNet, which does not pre-train CNNs with extra tracking datasets. For the sake of experimental fairness, the evaluation results of the trackers are provided by the VOT toolkit.

The expected average overlap plot evaluated on the VOT2016 benchmark is shown in Fig. 11. Fig. 11 demonstrates the EAO ranking, and the right-most tracker achieves the best performance. It is clear that our tracker performs favourably against other trackers used for comparison under the EAO metric. In addition, accuracy, robustness and EAO scores and rankings are summarized in Table 3, which illustrates that our tracker ranks first in robustness and EAO, and third in accuracy. As can be seen from Table 3, our tracker is superior to deep learning-based DNT, MDNet_N, SiamRN, and SiamAN, correlation filtering-based KCF and SAMF, as well as the trackers based on correlation filtering with deep features DeepSRDCF, deepMKCF, HCFT, and RFD_CF2 according to accuracy, robustness and EAO. The comprehensive second ranking tracker is VITAL, which is attributed to using generative adversarial learning to solve

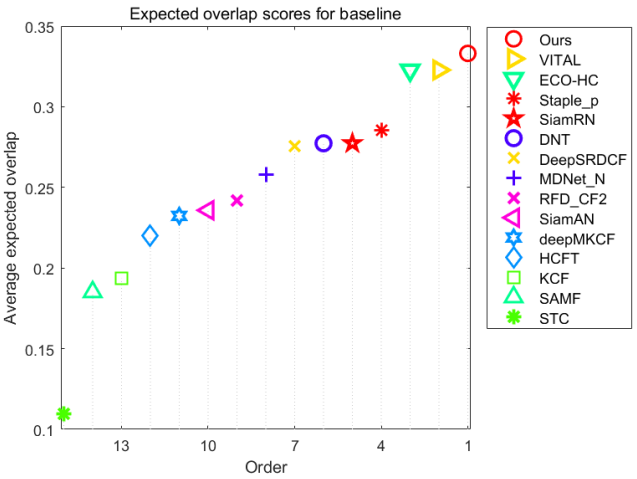


FIGURE 11. **Expected overlap scores for baseline experiment on the VOT2016 benchmark.**

the category imbalance problem. ECO-HC and Staple_p use hand-crafted features. In Table 3, the third ranking in robustness and EAO is occupied by ECO-HC, but it has low accuracy. In contrast, Staple_p has the highest accuracy, but the other two metrics are inferior. Specifically, our tracker significantly surpasses STC that is based on spatiotemporal

TABLE 3. Comparisons between our tracker and state-of-the-art trackers on the VOT2016 benchmark. Accuracy, robustness and EAO scores are listed. The first, second and third best results are marked by red, blue and green, respectively.

Tracker	Accuracy	Robustness	EAO
VITAL	0.556	0.275	0.323
ECO-HC	0.542	0.303	0.322
Staple_p	0.559	0.368	0.286
SiamRN	0.549	0.382	0.277
DNT	0.515	0.329	0.277
DeepSRDCF	0.529	0.326	0.276
MDNet_N	0.542	0.337	0.257
RFD_CF2	0.477	0.373	0.239
SiamAN	0.534	0.461	0.234
deepMKCF	0.544	0.422	0.232
HCFT	0.450	0.413	0.220
KCF	0.491	0.569	0.192
SAMF	0.507	0.587	0.185
STC	0.381	1.007	0.110
Ours	0.550	0.266	0.330

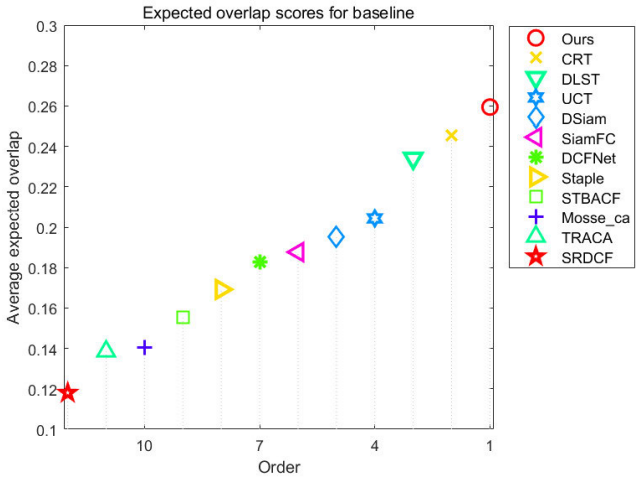


FIGURE 12. Expected overlap scores for baseline experiment on the VOT2017 benchmark.

features by 16.9% in accuracy and 22% in EAO. The experimental results on the VOT2016 show that compared with the other 14 trackers, our tracker performs best in robustness meanwhile it achieves high accuracy and EAO scores.

b: VOT2017 RESULTS

Fig. 12 shows the expected average overlap plot evaluated on the VOT2017 benchmark, where we compare our tracker with 11 state-of-the-art trackers provided by the VOT toolkit, including CRT [53], DLST [54], UCT [55], DSiam [18], SiamFC [17], DCFNet [20], Staple [9], STBACF [56], Mosse_ca [40], TRACA [35], and SRDCF [8]. The proposed tracker is at the right-most position in Fig. 12, which indicates that our tracker can obtain a superior performance in terms of the EAO metric. In addition, the evaluation results on the VOT2017 benchmark are reported in Table 4. As shown in Table 4, our tracker takes the first place in EAO, and the accuracy and robustness are also among the

TABLE 4. Comparisons between our tracker and state-of-the-art trackers on the VOT2017 benchmark. Accuracy, robustness and EAO scores are listed.

Tracker	Accuracy	Robustness	EAO
CRT	0.465	0.337	0.244
DLST	0.506	0.396	0.233
UCT	0.492	0.482	0.204
DSiam	0.512	0.654	0.196
SiamFC	0.503	0.585	0.187
DCFNet	0.470	0.543	0.182
Staple	0.530	0.688	0.169
STBACF	0.461	0.740	0.155
Mosse_ca	0.400	0.805	0.141
TRACA	0.424	0.857	0.137
SRDCF	0.490	0.974	0.119
Ours	0.511	0.398	0.257

top three. Although CRT performs well in robustness and has the second highest EAO score, its accuracy is relatively low, with a reduction of 4.6% compared with our tracker. In terms of the accuracy and robustness metrics, the performance of our tracker is similar to DLST, but our tracker has a gain of 2.4% in EAO. Moreover, DSiam and Staple achieve high accuracy, but the other two metrics are inferior to our tracker. Table 4 shows that our tracker performs better than the deep learning-based UCT and TRACA, Siamese network-based SiamFC and DCFNet, as well as the correlation filtering-based Mosse_ca and SRDCF in all three metrics. Among all the trackers used for comparison, we need to pay attention to STBACF, which is a variation of BACF. To improve the robustness of BACF, spatial and temporal regularization are incorporated into the original BACF formula. However, compared with our tracker, the performance of STBACF obtain a significant decrease in accuracy and EAO. The evaluation analysis on the VOT2017 indicates that our tracker has high accuracy, EAO scores and rare robustness, which demonstrates the effectiveness of the proposed tracker.

3) QUALITATIVE EVALUATION

Fig. 13 shows the tracking results of TRACA, ACFN, CFNet, SiamFC, SCT, Staple, SRDCF, CNN-SVM and our tracker on 10 challenging sequences from the OTB-2015. The SCT tracker does not perform well in all the presented sequences. Although SCT can adjust the distribution of attention according to different features and kernel types, the attention feature contains noise, which makes it impossible to distinguish the object and background for a long time. In contrast, ACFN performs well on IV (Human9) and OPR (DragonBaby) because of the re-detection mechanism, but it generally fails on other challenging sequences. SRDCF and Staple employ an empirical update method, which leads to limitations on OCC (Box, Girl2, Ironman). However, the Staple fuses multiple hand-crafted features, so it performs well on the DEF (Bolt2, Skating1) scenes. SiamFC has no model update mechanism, which causes the object to be lost in most scenes, but it can track the object accurately in FM (Human9) and

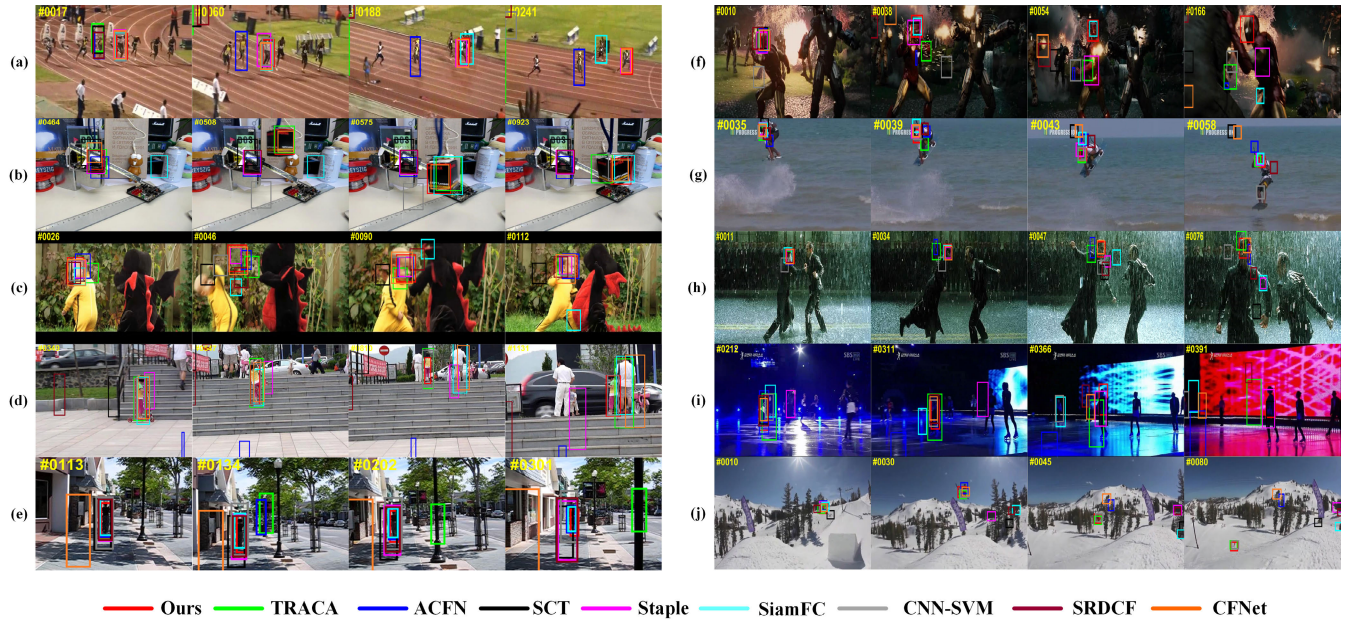


FIGURE 13. Qualitative evaluation of our tracker and comparison trackers on 10 challenging sequences. (a) Bolt2, (b) Box, (c) DragonBaby, (d) Girl2, (e) Human9, (f) Ironman, (g) KiteSurf, (h) Matrix, (i) Skating1 and (j) Skiing.

LR (Ironman) video sequences. CFNet adds a model update module based on SiamFC to deal with BC (Bolt2, Matrix) and OV (DragonBaby). In addition, when similar objects appear, such as Bolt2 and Skating1, most of the compared trackers lose the object. Although CNN-SVM adopts CNN features, it still fails on these challenging sequences, which can be attributed to the CNN features used from one convolutional layer, and the CNN features are sensitive to similar objects. TRACA compresses CNN features and adopts data augmentation, so it is very efficient for processing rotated sequences (KiteSurf, Skiing). TRACA does not further explore the potential of the model, which leads to poor performance on the DEF (Bolt2, Skating1) and MB (Human9) scenes. As shown in Fig. 13, the proposed tracker can accurately track most of the challenging sequences in the legend, which benefits from our tracker incorporating CNN features from numerous convolutional layers and introducing temporal and spatial networks to improve the tracker performance. Finally, we ensure model stability by updating the model. In general, our tracker is more precise and robust than the state-of-the-art trackers, especially in challenging scenarios.

VI. CONCLUSIONS

This paper proposes a deep ensemble object tracking algorithm based on temporal and spatial networks. We extract CNN features from numerous convolutional layers, redefine correlation filtering as one convolution layer, develop a branch network with temporal and spatial networks, use adaptive ensemble learning to combine multiple weak trackers into strong trackers, and finally obtain tracking results. Our algorithm solves the shortcoming that the correlation filtering algorithm cannot be trained end-to-end. Temporal and spatial networks can address the problem of fast motion and object appearance deformation, and effectively capture the object temporal and spatial information. In addition, adaptive ensemble learning is used to integrate the CNN features from numerous convolutional layers to further capture the object spatial information and semantic information, and the proposed new update method is adopted to ensure the stability of the model. Extensive experimental evaluations on large-scale benchmark datasets indicate that our tracker performs favourably compared with state-of-the-art trackers. However, for occlusion and low-resolution sequences, our algorithm still has limitations. Future work needs to start from this aspect to solve this problem.

REFERENCES

[1] Y. Wu, Y. Sui, and G. Wang, "Vision–based real–time aerial object localization and tracking for UAV sensing system," *IEEE Access*, vol. 5, pp. 23969–23978, 2017.

[2] G. Liu, S. Liu, K. Muhammad, A. K. Sangaiah, and F. Doctor, "Object tracking in vary lighting conditions for fog based intelligent surveillance of public spaces," *IEEE Access*, vol. 6, pp. 29283–29296, 2018.

[3] V. A. Prisacariu and I. Reid, "3D hand tracking for human computer interaction," *Image Vis. Comput.*, vol. 30, no. 3, pp. 236–250, Mar. 2012.

[4] 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.

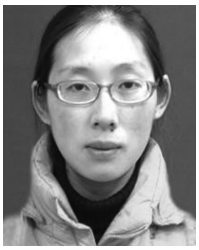
[5] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550.

[6] 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.

[7] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.* Nottingham, U.K.: BMVA Press, 2014.

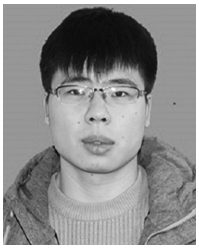
[8] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4310–4318.

[9] 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.
[10] 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.* Cham, Switzerland: Springer, 2016, pp. 472–488.
[11] 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.
[12] N. Wang and D. Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 809–817.
[13] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," 2015, *arXiv:1501.04587*. [Online]. Available: https://arxiv.org/abs/1501.04587
[14] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 597–606.
[15] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3119–3127.
[16] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 749–765.
[17] 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.* Cham, Switzerland: Springer, 2016, pp. 850–865.
[18] 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.
[19] 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.
[20] 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: https://arxiv.org/abs/1704.04057
[21] Z. Cui, S. Xiao, J. Feng, and S. Yan, "Recurrently target–attending tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1449–1458.
[22] H. Fan and H. Ling, "SANet: Structure–aware network for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 42–49.
[23] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Mar. 2006.
[24] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.
[25] N. Wang and D. Y. Yeung, "Ensemble-based tracking: Aggregating crowdsourced structured time series data," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1107–1115.
[26] 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.
[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[31] 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.
[32] K. Chaudhuri, Y. Freund, and D. J. Hsu, "A parameter-free hedging algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 297–305.
[33] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia (MM)*, 2015, pp. 689–692.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[35] J. Choi, H. J. Chang, T. Fischer, S. Yun, K. Lee, J. Jeong, Y. Demiris, and J. Y. Choi, "Context–aware deep feature compression for high–speed visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 479–488.
[36] 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.
[37] 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.
[38] 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.
[39] M. Kristan *et al.*, "The visual object tracking VOT2016 challenge results," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2016, pp. 777–823.
[40] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey, and G. Fernandez, "The visual object tracking VOT2017 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1949–1972.
[41] H. Nam, M. Baek, and B. Han, "Modeling and propagating CNNs in a tree structure for visual tracking," 2016, *arXiv:1608.07242*. [Online]. Available: https://arxiv.org/abs/1608.07242
[42] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M. H. Yang, "Fast visual tracking via dense spatio-temporal context learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2014, pp. 127–141.
[43] 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.
[44] B. Liu, Q. Liu, Z. Zhu, T. Zhang, and Y. Yang, "MSST–ResNet: Deep multi-scale spatiotemporal features for robust visual object tracking," *Knowl.-Based Syst.*, vol. 164, pp. 235–252, Jan. 2019.
[45] Z. Zhu, B. Liu, Y. Rao, Q. Liu, and R. Zhang, "STResNet_CF Tracker: The deep spatiotemporal features learning for correlation filter based robust visual object tracking," *IEEE Access*, vol. 7, pp. 30142–30156, 2019.
[46] 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.
[47] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "VITAL: VIsual Tracking via Adversarial Learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8990–8999.
[48] Z. Chi, H. Li, H. Lu, and M.-H. Yang, "Dual deep network for visual tracking," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 2005–2015, Apr. 2017.
[49] 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.
[50] R. Walsh and H. Medeiros, "Detecting tracking failures from correlation response maps," in *Proc. Int. Symp. Vis. Comput.* Cham, Switzerland: Springer, Dec. 2017, pp. 125–135.
[51] M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3038–3046.
[52] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Sep. 2014, pp. 254–265.
[53] K. Chen and W. Tao, "Convolutional regression for visual tracking," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3611–3620, Jul. 2018.
[54] L. Yang, R. Liu, D. Zhang, and L. Zhang, "Deep location–specific tracking," in *Proc. ACM Multimedia Conf. (MM)*, 2017, pp. 1309–1317.
[55] Z. Zhu, G. Huang, W. Zou, D. Du, and C. Huang, "UCT: Learning unified convolutional networks for real–time visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1973–1982.
[56] M. Kristan *et al.*, "The sixth visual object tracking VOT2018 challenge results," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–53.
[57] J. Nalepa, G. Mrukwa, and M. Kawulok, "Evolvable deep features," in *Proc. Int. Conf. Appl. Evol. Comput.*, 2018, pp. 497–505.



ZHAOHUA HU was born in 1981. She received the M.S. and Ph.D. degrees from the Nanjing University of Science and Technology, China. She was an Associate Professor with the Nanjing University of Information Science and Technology, China. Her main research interests include computer vision, deep learning, and visual tracking.

HUXIN CHEN was born in 1995. He is currently pursuing the master's degree with the Nanjing University of Information Science and Technology, China. His research directions are visual tracking and deep learning.



GAOFEI LI was born in 1993. He is currently pursuing the master's degree with the Nanjing University of Information Science and Technology, China. His research directions are visual tracking and machine learning.