Hierarchical Tracking by Reinforcement Learning-Based Searching and Coarse-to-Fine Verifying

Bineng Zhong, Bing Bai, Jun Li, Yulun Zhang, and Yun Fu, *Fellow, IEEE*

***Abstract*— A class-agnostic tracker typically consists of three key components, i.e., its motion model, its target appearance model, and its updating strategy. However, most recent top-performing trackers mainly focus on constructing complicated appearance models and updating strategies, while using comparatively simple and heuristic motion models that may result in an inefficient search and degrade the tracking performance. To address this issue, we propose a hierarchical tracker that learns to move and track based on the combination of data-driven search at the coarse level and coarse-to-fine verification at the fine level. At the coarse level, a data-driven motion model learned from deep recurrent reinforcement learning provides our tracker with coarse localization of an object. By formulating motion search as an action-decision problem in reinforcement learning, our tracker utilizes a recurrent convolutional neural network-based deep Q-network to effectively learn data-driven searching policies. The learned motion model can not only significantly reduce the search space but also provide more reliable interested regions for further verifying. At the fine level, a kernelized correlation filter (KCF)-based appearance model is adopted to densely yet efficiently verify a local region centered on the predicted location from the motion model. Through use of circulant matrices and fast Fourier transformation, a large number of candidate samples in the local region can be efficiently and effectively evaluated by the KCF-based appearance model. Finally, a simple yet robust estimator is designed to analyze possible tracking failure. The experiments on OTB50 and OTB100 illustrate that our tracker achieves better performance than the state-of-the-art trackers.**

***Index Terms*— Visual tracking, motion model, reinforcement learning, coarse-to-fine verification, correlation filter.**

I. Introduction

VISUAL tracking is one of the fundamental components in video understanding, and thus has various applications in surveillance, robotics, and autonomous driving. Typically, there are three key components in a class-agnostic tracker [1]–[6], i.e., its motion model, its target appearance model, and its updating strategy. Given the initial states of an object, a tracker firstly utilizes a motion model to search the likely states of an object. Then, based on its target appearance model, it calculates all candidate scores for the likely states, and thus derives the optimal target state. Finally, the target appearance model should be adaptively updated to capture complex time-varying appearances caused by occlusion, scale variations, non-rigid deformations, background clutters, and similar distractors.

Over the past decades, there have been various attempts to build a robust and real-time tracker from the three components put forth above. Interestingly, despite the vast progress driven by the OTB50 [1], OTB100 [2], and VOT [3] challenges, most state-of-the-art trackers focus on constructing more and more complicated appearance models to improve their discriminative power, and model updating strategies to alleviate the drifting problem. Representative trackers include conventional tracking-by-detection based trackers [1]–[6], correlation filter based trackers [7]–[17], and deep learning based trackers [18]–[32]. However, they neglect the third key component and use simple motion models which are brute-force, heuristic, hand-engineered, and independent of the video content to search the interested regions, such as, exhaustive or sliding windows based local search [1]–[6], mean shift based gradient search [33], and particle filtering based sampling search [34]. Consequently, the motion models may weaken the effectiveness of an elaborate appearance model and an updating strategy.

In this paper, inspired by how the human visual perception [35], [36] employs a hierarchical attention mechanism to immediately discard irrelevant input and actively focus on some fractions of interested regions, we propose a hierarchical tracker by reinforcement learning based searching and coarse-to-fine verifying. At the coarse level, a data-driven motion model learned from a deep recurrent reinforcement learning algorithm is used to estimate the translations and scale changes of the object, while discarding the majority of background. Some examples of the dynamic searching process of our data-driven model are illustrated in Fig.1. In contrast to brute-force or heuristic motion models, its goal is to leverage knowledge from large collections of training episodes to learn the policies on how objects are likely to move over time. Consequently, it can choose the optimal actions to significantly

Manuscript received January 14, 2018; revised August 20, 2018 and October 21, 2018; accepted November 23, 2018. Date of publication December 5, 2018; date of current version January 30, 2019. This work was supported in part by the Natural Science Foundation of China under Grants 61572205, 61802135, and 61728103, in part by the Natural Science Foundation of Fujian Province under Grant 2017J01113, and in part by NSF IIS under Award 1651902. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Khan M. Iftekharuddin. *(Corresponding author: Bineng Zhong.)*

B. Zhong and B. Bai are with the Department of Computer Science and Engineering, Huaqiao University, Xiamen 361021, China (e-mail: bnzhong@hqu.edu.cn; 1511314014@hqu.edu.cn).

J. Li, Y. Zhang, and Y. Fu are with the Department of Electrical and Computer Engineering, College of Computer and Information Science, Northeastern University, Boston, MA 02115 USA (e-mail: junl.mldl@gmail.com; yulun100@gmail.com; yunfu@ece.neu.edu).

Digital Object Identifier 10.1109/TIP.2018.2885238

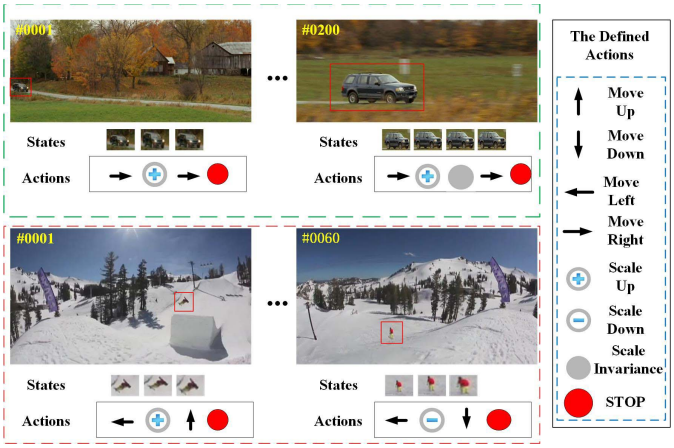


Fig. 1. Our deep recurrent reinforcement learning based data-driven motion model takes a sequence of actions to find the the translations and scale changes of an object in successive frames. In each frame, our motion model iteratively takes a candidate image patch centered at the previous estimated positions as input, and generates the probabilities of each action as the output. In each iteration, an optimal action is selected to move the target bounding box.

reduce the searching steps. At the fine level, we focus on densely yet efficiently evaluating a local region centered on the predicted location from the data-driven motion model. Since there are a large number of candidate samples existing in the local region, we use a kernelized correlation filter (KCF) based appearance model to efficiently and effectively verify the candidate samples by using circulant matrices and fast Fourier transformation. Finally, based on peak-to-sidelobe ratio (PSR) and median statistics, we design a simple yet robust estimator to analyze possible tracking failure. To sum up, the main contributions of this work are three-folds:

- We propose a robust hierarchical tracker that combines reinforcement learning based search and coarse-to-fine verification by naturally accentuating the advantages of each.
- Our tracker learns to move by a deep recurrent reinforcement learning algorithm which incorporates long short-term memory (LSTM) into a deep Q-network to fully exploiting the long-range time dependencies amongst states and actions. The learned motion model cannot only significantly reduce the search space, but also provide more reliable interested regions for further verifying.
- We design a simple yet robust estimator to provide supervision to an analysis of possible tracking failure and the adaption of a KCF based appearance model.

The rest of the paper is organized as follows: a brief overview of the related work is given in section II. Section III introduces our deep recurrent reinforcement learning based data-driven motion model. In section IV, our hierarchical tracker is described in detail. The experimental results are provided In section V. Finally, the paper is concluded in section VI.

II. Related Work

There is a huge amount of work on visual tracking which has been surveyed in [1]–[6] and [37]. This section provides a brief overview of some of the related trackers that are based on correlation filters and deep reinforcement learning.

A. Trackers Based on Correlation Filters

The trackers based on correlation filters (CF) [38] mainly focus on constructing a robust yet efficient appearance model by using the fast Fourier transform and circulant matrix. Since the pioneering work using the Minimum Output Sum of Squared Error (MOSSE) filter [7], the tracking performance of CF-based trackers has been persistently improved via a variety of extensions. Representative CF-based trackers include kernelized correlation filters [8], multi-channel features [8], [9], adaptive scale estimation [10], fusion of complementary learners [11], re-detection [13], long-term memory [14], multiple kernels [15], part-based representation [16] spatial regularization [39], [40], support vector machine [41], sparse coding [42], context and background-aware learning [12], [43]. However, one of the main limitations is that the above trackers use hand-crafted features to construct a shallow CF based appearance model. Consequently, they may fail in a complex scene.

Recently, to effectively use hierarchical and deep feature representations, a number of authors have successfully combined convolutional neural networks (CNNs) and CF for robust tracking. The trackers integrating CNN and CF can be roughly categorized into two classes: (1) the CF based trackers using pre-trained CNNs, and (2) the CF based trackers using end-to-end training with CNNs. The CF based trackers using pre-trained CNNs typically build CF based appearance models upon pre-trained CNN features from large-scale offline training data. Representative trackers include hierarchical CNN features [18], adaptive Hedge algorithm [19], DeepSRDCF [20], continuous convolution [44], multi-task correlation particle filter [17], and parallel tracking and verifying [24]. However, most of their trackers rely on the CNN features that are either empirically chosen or independently trained for a different task (e.g., image classification). In contrast, some authors explore the CF based trackers using end-to-end training with CNNs. Representative trackers include convolutional residual learning [21], end-to-end trained network for correlation filter (CFNet) [22], discriminant correlation filter based network (DCFNet) [23], and dynamic siamese network [45].

Instead of using CF, there are trackers that solely rely on CNNs. In [25] and [26], the CNN features are adaptively selected and sequentially updated to capture the target appearance variations at test time, respectively. In [27] and [28], based on the pre-trained CNN features, a dual network and a per-object classifier are online constructed for visual tracking, respectively. However, online updating the pre-trained CNN features is not only time-consuming, but also deteriorates tracking performance due to lack of online training data. To improve the processing speed, Held *et al.* [29], Bertinetto *et al.* [30], Gordon *et al.* [31], and Tao *et al.* [32] utilize Siamese networks for visual tracking, in which a generic tracker is constructed by using solely large-scale off-line training data and online update is not required.

However, the above trackers mainly focus on construct a discriminative and robust target appearance model. Consequently, they may neglect the importance of motion models, and thus use brute-force and heuristic motion models to provide the interested regions.

B. Trackers Based on Deep Reinforcement Learning

The essence of reinforcement learning (RL) [46] is learning good policies through trial-and-error interaction. Recently, with the rise of deep learning, deep reinforcement learning (DRL) [47] has gained considerable attention and been successfully applied to a variety of applications, e.g. visual semantic planning [48], object detection [49], the games of Atari and Go [50]–[52] etc. Inspired by the success of DRL, several works have viewed visual tracking as a decision-making process to address the three key components of a tracker from different aspects.

The first family of DRL based trackers focus on constructing the target appearance models. Choi *et al.* [53] use a DRL algorithm for constructing a template based appearance model, in which suitable templates are adaptively selected for tracking a given frame. Zhang *et al.* [54] focus on constructing a target appearance model by using a recurrent convolutional neural network (R-CNN) which can encode both spatial and temporal constraints. To address the real-time requirement without losing accuracy in a deep learning based tracker, Huang *et al.* [55] use a decision-making process to formulate the adaptive tracking problem, in which DRL is used to learn an early-stopping decision policy for speeding up a siamese network based tracker [30]. Although the proposed early-stopping tracker drastically reduce the computation cost, the early layers in the feature cascade still have to be applied exhaustively over all image regions. In other words, they still adopt a brute-force motion model to provide candidate regions. In [56], visual tracking is formulated as a partially observable decision-making process, in which a policy is learned from interactive streaming videos to decide where to look in an incoming frame, when to update a target appearance model, and when to re-initialize it. Though such decisions are learned with DRL, they ignore the long-range temporal information. Moreover, their approach requires a complex streaming interactive training mechanism.

The most similar work to ours is the work of Yun *et al.* [57]. They propose a tracker which actively searches an object via a sequence of actions controlled by an Action-Decision network (ADNet). The ADNet is not only offline pre-trained, but also updated in an online manner which significantly increases the calculational cost during the tracking process. In contrast, our data-driven motion model can take full advantage of the long-range temporal information by incorporating LSTM into our deep current reinforcement learning algorithm. Moreover, our data-driven motion model is not online updated, which significantly reduces the calculational cost. More importantly, our hierarchical tracker can take full advantage of the reinforcement learning based search and coarse-to-fine verification by naturally accentuating the advantages of each.

III. Our Data-Driven Motion Model

In this section, we firstly formulate a motion prediction problem in visual tracking as a deep reinforcement learning problem. Then, we design our recurrent CNN based deep Q-network for learning policies. Next, we describe how to train the model in details.

A. Problem Formulation and Learning Setting

As shown in Fig.1, our goal is to find a sequence of actions that progressively moves a tracker from its current state (i.e., locations and scales) to localize a target object at each frame. We develop a DRL model that takes as input a candidate image patch centered at the estimated positions of the current state. The output of the DRL model is the probabilities of each action, such as horizontal moves (e.g., left or right), vertical moves (e.g., up or down), scale changes (e.g., bigger, smaller, or invariance), and stop. According to the predicted action, the tracker is moved to a newly state. Then, the tracker decides how to choose next action from the newly state. The searching procedure iteratively continues until the target object is localized.

More specifically, we formulate the problem of motion prediction as a Markov decision process (MDP) in a general RL setting, which can be specified by several key components: action space A, state space S, and a reward function R. Let $f \in \{1, \ldots, F\}$ and $t \in \{1, \ldots, T_f\}$ denote the number of frames in a video and the number of iterated time steps at the f^{th} frame, respectively. Given an observation image I_f and current state $s_{t,f} \in S$ at t^{th} time step, an agent performs an action $a_{t,f} \in A$, which is determined by a policy function $\pi : S \rightarrow A$. The action $a_{t,f}$ can be chosen via a deterministic or stochastic manner. After taking the action $a_{t,f}$, the agent observes the next state $s_{t+1,f}$, and thus receives a scalar reward $r_{t,f}$ reflecting how well the next state (i.e., the estimated bounding box) covers the target object. The iterative process continues until the agent reaches a terminal state. By maximizing the discounted sum of expected future rewards, the agent learns the best policy π to take actions, and thus can effectively search the target object in an efficient manner. In the following sections, we omit the subscript f for more clear descriptions on the MDP at each frame.

1) Action Space: As shown in Fig.1, we consider eight types of actions in our action set A: two horizontal moves (e.g., left or right), two vertical moves (e.g., up or down), three scale changes (e.g., bigger, smaller, or invariance), and one stopping action to terminate search. Each action is represented by an 8-dimensional one-hot vector, where the value corresponding to the taken action is set as 1, and other ones are zero. Inspired by Caicedo and Lazebnik [49], any of these actions makes a combined horizontal, vertical and diagonal change to the bounding box by a factor of 0.2 relative to its current size. The aspect ratio of the target object is maintained when the scaling actions are performed.

2) State Space: At t^{th} time step, the state $s_t \in \Re^{224\times224\times3}$ is an observed image patch, which is associated with a bounding box and localized by the coordinates of its upper left and lower right corners: $o_t \in \Re^4$. Based on o_t, the observed image patch

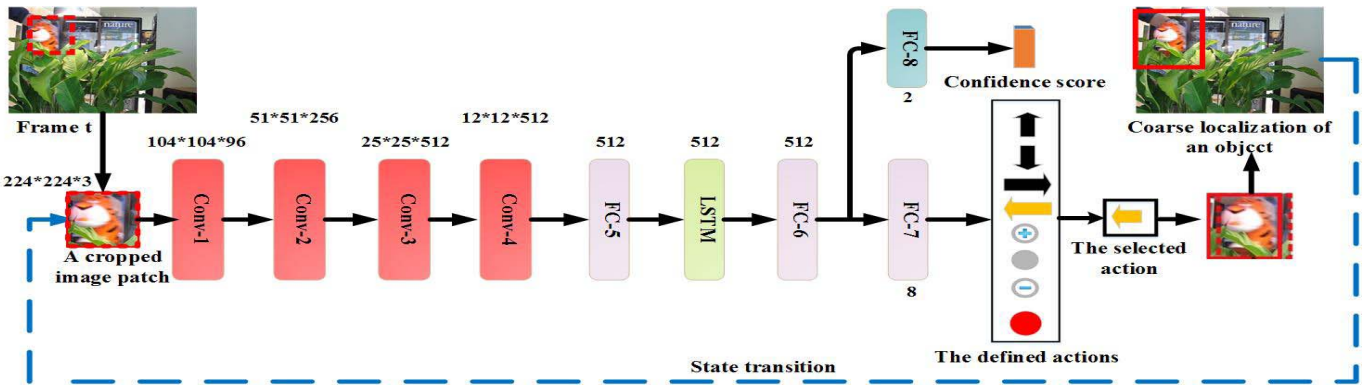


Fig. 2. The architecture of our deep recurrent reinforcement learning based deep Q-network for motion prediction. As shown in the figure, the tracker decides which action is to choose and controls the tracker to move.

is cropped, and then resized to match the input size of our DRL network (i.e., 224×224)

3) Reward Function: The reward function R reflects the tracking accuracy improvement by taking an action a_t under a state s_t. In this paper, we adopt a simple yet effective function to measure the accuracy, i.e., the Intersection-over-Union (IoU) between the predicted bounding box o_t and ground-truth bounding box g_t: $IoU = area(o_t \cap g_t)/area(o_t \cup g_t)$. Then, the reward function R is defined by the followings:

$$R = \begin{cases} +IoU & IoU \geq \rho \\ -IoU & IoU < \rho, \end{cases} \tag{1}$$

where ρ is an IoU threshold. The $'+'$ or $'-'$ IoU reward implicitly encourages a better action or penalizes a large number of actions by avoiding the agent being trapped in the endless refinement of a single step and promoting the search for new directions.

B. Our DRL Model for Learning Policy

In this subsection, we describe how to learn the optimal policy function $\pi : S \rightarrow A$ to select actions via DRL. Typically, based on Bellman equations, a state-action value function $Q^{\pi}(s_t, a_t)$ is recursively defined to return an expected future reward for each action a_t:

$$Q^{\pi}(s_t, a_t) = R(s_t) + \gamma \, max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1}), \tag{2}$$

where $Q^{\pi}(s_{t+1}, a_{t+1})$ and $\gamma \in [0, 1]$ are the future reward and the discount factor, respectively. In the traditional RL methods, the optimal state-action value function Q^{π} can be iteratively learned via Q-learning and the optimal action to take can be estimated from the followings:

$$a_t^* = argmax_a Q^{\pi}(s_t, a). \tag{3}$$

In this paper, based on deep Q-learning [50], we design a new recurrent CNN based deep Q-network (DQN) as a non-linear function approximator for π. The detailed structure of our recurrent CNN based DQN is shown in Fig.2. As shown in Fig.2, the policy π is learned by our recurrent CNN based DQN, which takes the current state representation s_t as input and generates the probabilities of each action a_t as the output. The network consists of four convolutional layers, four fully- connected layers, and one LSTM layer. Each convolutional layer is followed by a max-pooling layer. The first two fully-connected layers are followed by ReLU and BN regularization. The last two fully-connected layers predict the probabilities of each action and confidence score of the current state, respectively. The filter sizes in the four convolutional layers are set as $7 * 7$, $5 * 5$, $3 * 3$, and $3 * 3$, respectively. Meanwhile, the strides in the four convolutional layers are set as 2. Please note that, we incorporate the LSTM layer to fully capture the long-range history information after the first fully connected layer. Such design enables long-range knowledge transfer across states and actions, and therefore allows the model to learn to remember, forget, and ignore information about the appearance and motion of an object.

C. Training Our DRL Model

To effectively pre-train our recurrent CNN based DQN, we collect training episodes from the VOT-2016 data set [3] which has 60 short video sequences containing various objects in challenging scenes. Please note that the video sequences overlapping with OTB50 [1] and OTB100 [2] are excluded in the training episodes. There are 10,000 epochs in our training setting and each epoch contains 100 episodes. Each episode is a randomly sampled video clip with length of 200 from the VOT-2016 data set. In this paper, we adopt a variant of policy-based RL method to train our model [58] since it has better convergence properties and capability of learning stochastic policies.

The parameters θ of our recurrent CNN based DQN are randomly initialized. At an iterated training stage, a state representation s_t from the current frame is firstly fed into our recurrent CNN based DQN. Then, our recurrent CNN based DQN calculates the probabilities $p(a|s_t; \theta)$ of each action. Furthermore, we adopt a $\epsilon - greedy$ strategy to choose the current action via the following equation:

$$a_t^* = \begin{cases} a_t = argmax_a p(a|s_t; \theta) & 1 - \epsilon \\ Random \ choose \ an \ action \ in \ (A - a_t) & \epsilon. \end{cases} \tag{4}$$

The optimal action estimated by our model is selected by probability $1 - \epsilon$, while one of the resting actions is selected

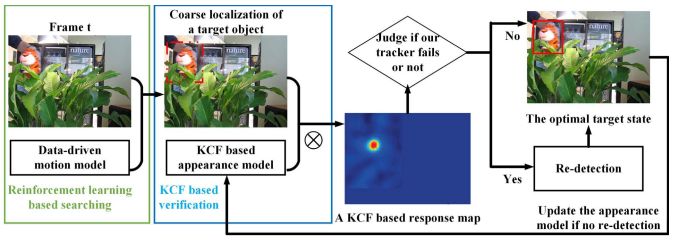


Fig. 3. Overview of our hierarchical tracker based on the combination of data-driven search and coarse-to-fine verification.

by probability ϵ. This could make all the actions have an opportunity to be chosen. Consequently, there is a different path to get the optimal policy in each iteration. Based on the selected action, the newly positions and scales of the target are estimated. Meanwhile, the corresponding reward r_t is computed using the Eq.(1). This tracking simulation process is iterated until the end of the training episode. Finally, by maximizing the discounted sum of expected future rewards, the parameters θ of our recurrent CNN based DQN are updated by a stochastic gradient descent algorithm [58] as the followings,

$$\nabla\theta \propto \sum_{t=1}^{T} \frac{\partial log p(a_t|s_t;\theta)}{\partial\theta} r_t. \tag{5}$$

IV. Our Hierarchical Tracker

In this section, we present our hierarchical tracker based on the combination of data-driven search and coarse-to-fine verification. The basic idea is to firstly employ a data-driven motion model to immediately discard irrelevant backgrounds, and then densely yet efficiently evaluate on some fractions of interested regions by a KCF based appearance model. The workflow of our tracker is illustrated in Fig.3. In the followings, we will briefly explain our framework. Then, we describe the details for each component in our framework.

More specifically, our hierarchical tracker works as follows: given a new frame, a context image patch centered on the previous target state is firstly cropped. Then, based on the image patch and our data-driven motion model, a coarse state of the target object is estimated by taking a sequence of actions to find the translation and scale changes of an object in current frame. Furthermore, through using of circulant matrices and fast Fourier transformation, a KCF based appearance model is used to densely yet efficiently verify a large number of candidate samples in a local region centered on the coarse localization of the object. Based on the confidence map generated by the KCF based appearance model, a simple yet robust estimator is used to analyze possible tracking failure. If the tracking failure is not detected by the estimator, the optimal target state is determined by finding the maximum position in the response map. Meanwhile, the KCF based target appearance model is updated to effectively capture target appearance changes. Otherwise, a re-detection mechanism is performed to search the missing object. The tracking process is repeated until the end of the video.

A. A KCF Based Appearance Model

We choose the HCF tracker [18] as a basic appearance model of our tracker. To simultaneously use semantic information and low-level details for constructing an appearance model, HCF [18] firstly apply KCF [8] to each convolutional layer from a pre-trained VGG network. Then, a coarse-to-fine searching schema is used to fuse the three KCF response maps.

In the standard KCF [8], the goal is to find the classifier $g(z) = \langle\omega, \Phi(z)\rangle$ trained on the feature $\Phi(x_{m,n})$ of each training image patch $x_{m,n}$ and the expected response $y_{m,n}$ which is a Gaussian function peaked at the center. The desired filter ω can be obtained by optimizing the following Ridge regression loss function:

$$\min_{w} \sum_{m,n} (\langle\Phi(x_{m,n}), \omega\rangle - y_{m,n}) + \lambda\|\omega\|^2, \tag{6}$$

where $\Phi(x)$ is the mapping to a kernel space, and $\lambda \geq 0$ is the regularization parameter which controls overfitting and simplifies the model. After a nonlinear transform, the classifier can be denoted by other form $g(z) = \sum_{m,n} \alpha_{m,n} K(x_{m,n}, z)$, where $K(\cdot,\cdot)$ is a kernel function and $\alpha_{m,n}$ is the variables under optimization. The solution is derived as:

$$\widehat{\alpha} = (\widehat{k^{x_{1,1}x_{1,1}}} + \lambda)^{-1}\widehat{y}, \tag{7}$$

where $\widehat{\ }$ denotes the discrete Fourier transform, $\widehat{k^{x_{1,1}x_{1,1}}}$ is the first row of the kernel matrix $K = C(k^{x_{1,1}x_{1,1}})$, and $C(\cdot)$ is a circulant matrix.

During the tracking phase, we firstly crop a search patch z in the incoming frame. Then, the translation can be estimated by searching the maximum value of the correlation response map $g(z)$:

$$g(z) = G^{-1}(\hat{g}(z)) = G^{-1}(\hat{k}^{\bar{x}z} \odot \hat{\alpha}), \tag{8}$$

where $\bar{x}$ denotes the current target appearance. Given the newly state of the target object, the KCF based appearance model can be adaptively updated as follows if necessary:

$$\alpha^t = (1-\eta)\alpha^{t-1} + \eta\alpha^t, \tag{9}$$

where η is a learning rate. Please refer to [8] and [18] for more details.

B. A Simple Yet Robust Estimator for Failure Detection

To effectively capture complicated appearance variations while activating a re-detection mechanism after occlusion, we propose a simple yet robust estimator for tracking failure detection. Specifically, we forecast the existence of tracking failure based on the correlation response maps. Firstly, given the KCF response map c_t at time t, we use a peak absolute value $v(c_t)$ and the peak-to-sidelobe ratio (PSR) [7] $p(c_t)$ to jointly measure its tracking confidence:

$$\beta_t = v(c_t) \cdot p(c_t). \tag{10}$$

Then, we calculate the median β_t^m of $\{\beta_i\}_{i=t}^{t-N}$ over the recent N confidently tracked frames. Finally, our estimator is defined as the ratio between β_t^m and β_t:

$$E = \beta_t^m / \beta_t. \tag{11}$$

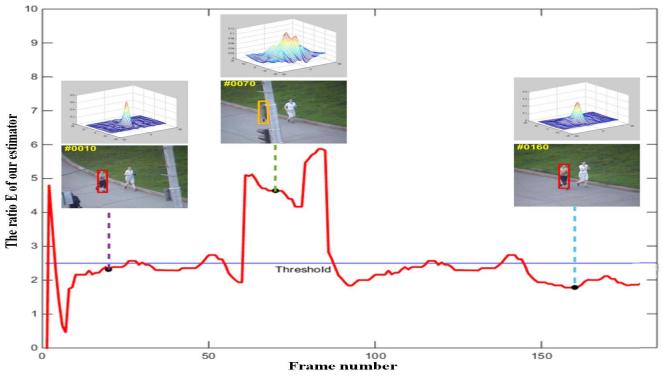


Fig. 4. The curve of the proposed estimator E for the Jogging sequence. The values above a predefined threshold indicate the possible tracking failure caused by occlusion or dramatic appearance variations.

If the ratio E is large than a threshold τ, our tracker is considered to fail in current frame and a corresponding re-detection mechanism is activated. In Fig.4, it can be seen that the proposed estimator E is an efficient confidence metric. The value of the proposed estimator E becomes large when a target object is occluded.

C. A Re-Detection Mechanism

A re-detection mechanism is activated to search the missed target object once the tracking failure is detected. The main idea is to use our data-driven motion model as a generic yet effective detector which is pre-trained from the offline training episodes. This allows our tracker to not only limit the drifting problem, but also keep adaptive to appearance changes. Specifically, we firstly generate a set of random samples with a Normal distribution whose mean value is the predicted position of the KCF based appearance model. Then, we use our deep recurrent reinforcement learning based data-driven motion model to calculate the confidence scores of the random samples. Finally, a sample with the maximum confidence score is selected to be the re-detected target position.

V. Experiments

In this section, we firstly present the experimental settings. Secondly, we compare our hierarchical tracker with other trackers on OTB50 [1] and OTB100 [2], respectively. Moreover, we perform an ablation study on several components of our hierarchical tracker. Finally, we evaluate how the parameter variations affect our tracker.

A. The Experimental Settings

We have implemented our hierarchical tracker in Tensorflow on a computer with a GTX-1060 GPU. The running speed of our hierarchical tracker is 6.3 fps. Empirically, the parameters of the IoU threshold ρ, discount factor γ, and ϵ are set to 0.5, 0.95 and 0.01, respectively. For the KCF based appearance model, we use the same parameters as the authors have given in their paper [18] for all of the experiments. The learning rate η in the KCF based appearance model is set to 0.01.

TABLE I
The Parameter Values of Our Hierarchical Tracker

ρ	γ	ϵ	τ	η
0.5	0.95	0.01	2.6	0.01

To train our DRL model, the number of epochs is set to 30. The learning rate in training DRL model is initially set to 0.001, while its decay is set to 0.1 times every 10 epochs. For the re-detection, the threshold τ for the ratio E is set to 2.6, and we draw 128 random samples centered around the predicted position of the KCF based appearance model. The ratio of the frames activating the re-detection mechanism to the all frames is around 7.6%. The corresponding parameters are shown in TABLE I.

Datasets: The OTB50 [1] data set contains 50 image sequences which are annotated with eleven attributes, including background clutter, scale variation, pose change, partial occlusion and so on. The OTB100 [2] data set has 100 image sequences and is the extension of the OTB50. Both OTB50 and OTB100 adopt precision and success metrics to evaluate tracking performances, i.e., center location error and bounding box overlap ratio. Specifically, the precision metrics are measured as the center location errors between ground-truth bounding boxes and predicted ones. In the precision plots, the x-axis denotes the distance thresholds on center location error, while the y-axis denotes the percentages of correctly predicted frames per threshold. The trackers are typically ranked by the precision under a distance threshold of 20 pixels. On the other hand, the success metrics measure the IoU of ground-truth bounding boxes and predicted ones. In the success plots, the x-axis denotes the overlapping thresholds, while the y-axis denotes the percentages of correctly predicted frames per threshold. The area under the curve (AUC) is used to rank the trackers.

B. Comparative Validation on OTB50 and OTB100

1) Quantitative Results: Based on the evaluation protocol of the OTB50 and OTB100, we compare our hierarchical tracker with fifteen state-of-the-art trackers which can be roughly categorized into three types: (I) The KCF based trackers without deep features, e.g., spatially regularized discriminative correlation filter with decontamination (SRDCFdecon) [40], channel and spatial reliability based discriminative correlation filter (CSRDCF) [9], kernelized correlation filter (KCF) [8], and sum of template and pixel-wise learners (Staple) [11]; (II) The deep learning based trackers, e.g., convolutional residual learning based tracker (CREST) [21], hierarchical convolutional feature based tracker (HCF) [18], hedged deep tracker (HDT) [19], fully convolutional network based tracker (FCNT) [25], fully-convolutional siamese network based tracker (SiamFC) [30], Multi-task correlation particle filter based tracker (MCPF) [17], and spatially regularized discriminative correlation filters with Convolutional features (DeepSRDCF) [20]; (III) The reinforcement learning based trackers, e.g., action-decision network based tracker (ADNet) [57], deep reinforcement learning based

TABLE II

THE QUANTITATIVE COMPARISON RESULTS ON OTB50 [1] OBTAINED BY OUR HIERARCHICAL TRACKER AND OTHER REINFORCEMENT LEARNING BASED TRACKERS

Tracker	AUC	Precision	Speed (fps)	Device
DRLT [54]	0.543	0.664	**45**	GTX 1080
RDT [53]	0.654	—	43	GTX TITAN X
EAST [55]	0.638	—	23	GPU Unkonwn
ADNet [57]	0.659	0.903	2.9	GTX TITAN X
Ours	**0.681**	**0.921**	6.3	GTX 1060

tracker (DRLT) [54], reinforced decision making based tracker (RDT) [53], and early-stopping tracker (EAST) [55].

Fig.5 shows the precision and success plots on OTB50 and OTB100. Please note that there are some different compared trackers in OTB50 and OTB100 due to the raw results of some trackers are unavailable for both OTB50 and OTB100. Compared with HCF [18] that only uses the basic KFC based appearance model, we observe improvements of +3% and +7.6% in the precision and AUC on OTB50, respectively. Meanwhile, we also observe improvements of +5.7% and +9.1% in the precision and AUC on OTB100, respectively. These results indicate that our learned data-driven motion model can provide more reliable interested regions for coarse-to-fine verifying using the basic KFC based appearance model. Compared with ADNet [57] that only uses reinforcement learning for motion prediction, our hierarchical tracker outperforms ADNet by 1.8% and 2.2% in the precision and AUC on OTB50, respectively. On OTB100, our hierarchical tracker achieves 89.4% and 65.1% in the precision and AUC, respectively. ADNet [57] achieves 88.1% and 64.6% in the precision and AUC on OTB100, respectively. HCF [18] achieves 83.7% and 56.0% in the precision and AUC on OTB100, respectively. Compared to the results of ADNet [57] and HCF [18], we also observe significant improvements on OTB100. These results are consistent with our expectation that reinforcement learning based search and coarse-to-fine verification can be effectively combined in a hierarchical manner by naturally accentuating the advantages of each. Compared to other top-performing trackers with deep features (e.g., CREST [21], MCPF [17], and DeepSRDCF [20]), our hierarchical tracker achieves promising results. The reason may be these trackers mainly focus on constructing complicated appearance models with, while use relatively simple motion models to search the interested regions.

In Table II, we additionally compare with other four reinforcement learning based trackers (e.g., DRLT [54], RDT [53], EAST [55], and ADNet [57].) according to the tracking results reported in their papers. Our tracker outperforms these trackers because they either focus on constructing appearance models or motion model. The robustness of our tracker lies in the systematically fusion of three key components (i.e., a reinforcement learning based motion model, a KCF based target appearance model, and an estimator based updating strategy) in a hierarchical manner.

2) Attribute-Based Evaluation: Although our hierarchical tracker achieves promising tracking performance in most scenarios, we further analyze its performance under eleven

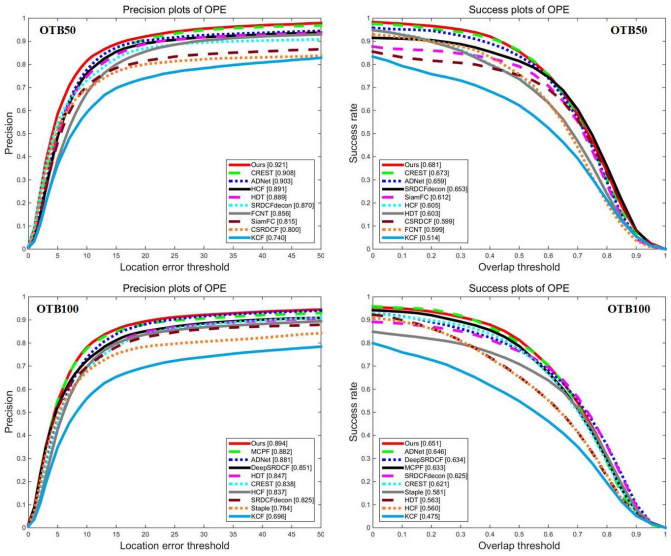


Fig. 5. The precision and success plots on OTB50 [1] and OTB100 [2] using one-pass evaluation (OPE).

attributes annotated in OTB50 [1]. We only show the results on six attributions in Fig.6 due to the space limitation. As shown in Fig.6, our hierarchical tracker is good at handling most attributes. On the attribute of motion blur, Our tracker and ADNet [57] achieve the first and second best performance, respectively. The reason is that the motion model is explicitly constructed by reinforcement learning in the two trackers. On the attribute of occlusion, we observe improvement of 2.3%, 3.4%, and 5.2% in the precision compared with CREST [21], HCF [18], ADNet [57], respectively. The benefit of our tracker comes from the robust estimator designed to analyze possible tracking failure.

3) Qualitative Results: Fig.7 shows the qualitative comparison results on six challenging sequences (i.e., motorrolling, walking2, skiing, lemming, ironman, and singer2) which contain fast motion, occlusion, illumination changes, pose and scale variations. For example, the lemming sequence on the forth row contains occlusion. The serious occlusion results in the failure of most compared trackers. In contrast, our hierarchical tracker can accurately locate the target object due to a re-detection mechanism is activated to search the missed target object once the tracking failure is detected. Moreover, the ironman sequence on the firth row suffers from significant illumination variations and the abrupt movement of a target object. Our tracker can satisfactorily handle such large sudden changes due to it effectively adopt a coarse-to-fine verification. According to the overall results, our hierarchical tracker exhibits robustness in challenging sequences which contains fast motion, occlusion, and scale variation.

C. Diagnostic Analyses of Our DRRL Tracker

In this subsection, we provide an ablation study on our hierarchical tracker. Firstly, we show how much the specific algorithmic components contribute to the overall tracking performance. Secondly, we investigate the effectiveness of sequential actions learned by our hierarchical tracker.

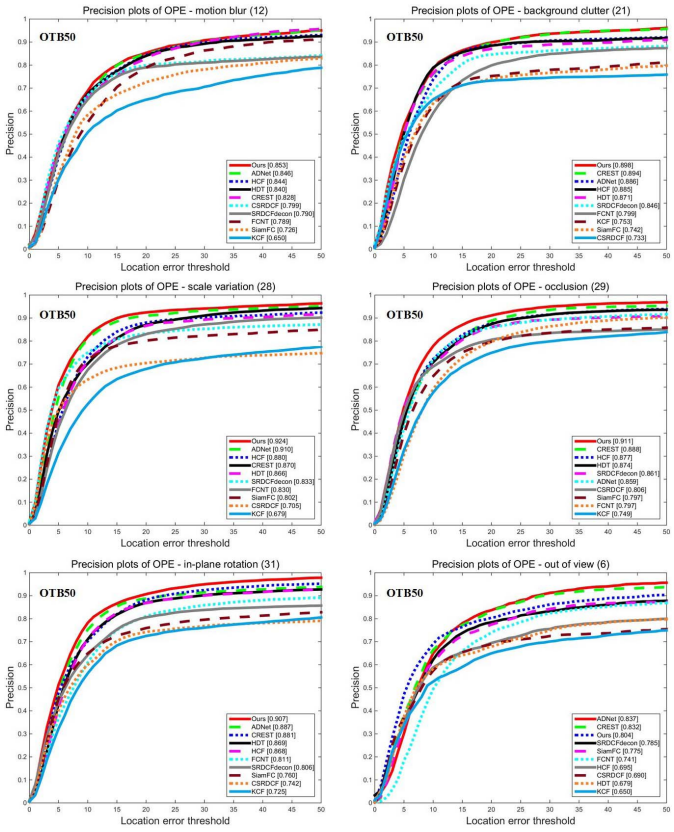


Fig. 6. The precision plots for six different tracking attributes: motion blur, background clutter, scale variation, occlusion, in-plane rotation, and out-of-view. Our hierarchical tracker shows better performance.

Fig. 7. Qualitative comparison on six image sequences from OTB50 [1], i.e., motorrolling, walking2, skiing, lemming, ironman, and singer2.

1) The Contributions From Specific Algorithmic Components: There are several important components in our hierarchical tracker, i.e., a deep recurrent reinforcement learning (DRL) based motion model with LSTM (denoted as DRL+LSTM), an HCF based appearance model (denoted as HCF), and a re-detection mechanism (denoted as

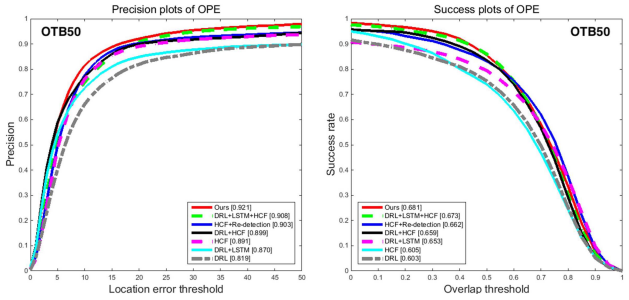


Fig. 8. The precision and success plots on OTB50 [1] for different variants of our hierarchical tracker.

Re-detection). To gauge their contributions to the proposed hierarchical tracker, we evaluate six variants of our hierarchical tracker by gradually removing each component and validate their performance on OTB50. The first degraded tracker is denoted as DRL+LSTM+HCF, in which the Re-detection is removed from our hierarchical tracker. The second degraded tracker is denoted as DRL+LSTM, in which both the HCF and Re-detection are removed from our hierarchical tracker. The third degraded tracker is denoted as DRL, in which the LSTM, HCF and Re-detection are removed from our hierarchical tracker. The fourth degraded tracker is denoted as HCF+Re-detection, in which the DRL+LSTM are removed from our hierarchical tracker. The fifth and sixth degraded tracker is composed of DRL+HCF and HCF, respectively. As shown in Fig.8, the tracking performance gradually decreases when we remove the Re-detection, HCF, and LSTM from our tracker step by step. In addition, one major advantage of our hierarchical tracker is the ability to address the long-range information aggregation by incorporating LSTM in a deep reinforcement learning based motion model. According to Fig.8, the degraded trackers without LSTM is worse than the trackers with LSTM. In other words, DRL is worse than DRL+LSTM while DRL+HCF is worse than DRL+LSTM+HCF. The compared results clearly verify the advantage that the LSTM make the most contributions to our tracker. This is because that our tracker can effectively use LSTM to remember and utilize the historical information to encode the motion model. Finally, our tracker is better than the degraded trackers (i.e., HCF and HCF+Re-detection) without the data-driven search at the coarse level. This clearly verifies the advantage of coarse-to-fine verification.

2) The Effectiveness of Sequential Actions: In this subsection, we show how the sequential actions learned by our hierarchical tracker are effectively used to improve the tracking performance. In Fig.9, we visualize some representative actions and their probabilities that are provided by our tracker from the carScale sequence. As expected, though the motion variations are challenging, we can predict the most likely actions. Moreover, this case study also shows our tracker is robust to scale variations. Furthermore, compared with brute-force or heuristic searching strategies (e.g., sliding window methods), most of the frames requires only 3 or 4 actions to pursue a target object in each frame. It shows that our tracker achieves fast and coarse localization of a target

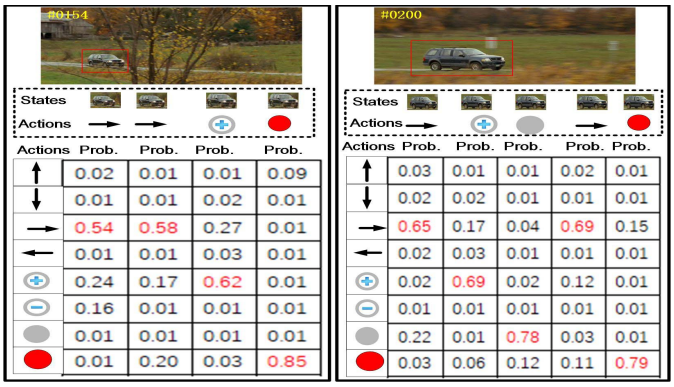


Fig. 9. Illustration of the action selection process on the carScale sequence from OTB50 [1]. Red indicates the highest probability of the optimal action in one iteration.

TABLE III

PRECISION UNDER DIFFERENT PARAMETER VALUES FOR THE OTB50. WHILE ONE PARAMETER IS VARIED, OTHER PARAMETERS ARE KEPT FIXED AT THE VALUES GIVEN IN TABLE I

ρ	0.3	0.4	**0.5**	0.6	0.7
Precision↑	0.871	0.898	0.921	0.909	0.882
γ	0.940	0.945	**0.950**	0.955	0.960
Precision↑	0.912	0.917	0.921	0.918	0.910
τ	2.0	2.5	**2.6**	2.7	2.8
Precision↑	0.883	0.916	0.921	0.919	0.899

object. The percentage of the frames requiring more than five actions is only around 3.6%. In conclusion, our tracker can learn an efficient data-driven motion model to improve the performance.

D. Sensitivity to Parameters

Due to the proposed hierarchical tracker contains some parameters, there naturally arises the following questions: (1) how to choose the parameters; and (2) how the small changes of the parameter values will affect the performance of the proposed hierarchical tracker. To answer the questions, we check the precision for different parameter settings. We only change one parameter at a time due to relatively many combinations. Table III shows the experimental results from OTB50. It is obvious that, for all parameters, a good value can be chosen across a wide range of values. This property also show that a good set of parameter values is easily obtained. Moreover, the experiments have illustrated that the good parameters for an image sequence often work well also for other image sequences (see Fig.5, Fig.6 and Fig.7).

VI. CONCLUSION

In this paper, we have proposed a hierarchical tracker that learns to move and track based on the combination of data-driven search and coarse-to-fine verification. The main contributions are made by the work: (1) We have developed a novel solution which integrates motion, appearance, and re-detection in a hierarchical fashion by naturally accentuating the advantages of each; (2) By incorporating LSTM into a deep recurrent reinforcement learning algorithm, we have developed a fully trainable data-driven motion model that cannot only significantly reduce the search space, but also provide more reliable interested regions for further verifying; (3) We have designed a simple yet robust estimator to analyze possible tracking failure and uncertainty. Encouraging results on OTB50 and OTB100 demonstrate the effectiveness and efficiency of our hierarchical tracker.

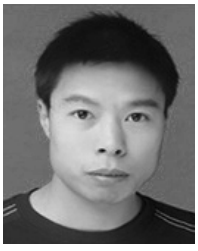
REFERENCES

[1] 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.
[2] 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.
[3] M. Kristan *et al.*, "The visual object tracking VOT2015 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, Dec. 2015, pp. 1–23.
[4] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. van den Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 4, p. 58, Sep. 2013.
[5] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.
[6] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.
[7] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 2544–2550.
[8] 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.
[9] A. Lukežič, T. Vojíř, L. Čehovin, J. Matas, and M. Kristan. (2016). "Discriminative correlation filter with channel and spatial reliability." [Online]. Available: https://arxiv.org/abs/1611.08461
[10] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, Nottingham, U.K., Sep. 2014, pp. 1–11.
[11] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1401–1409.
[12] 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.* New York, NY, USA: Springer, 2014, pp. 127–141.
[13] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5388–5396.
[14] 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.*, Jun. 2015, pp. 749–758.
[15] M. Tang and J. Feng, "Multi-kernel correlation filter for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 3038–3046.
[16] Y. Li, J. Zhu, and S. C. H. Hoi, "Reliable patch trackers: Robust visual tracking by exploiting reliable patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 353–361.
[17] T. Zhang, C. Xu, and M.-H. Yang, "Multi-task correlation particle filter for robust object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jul. 2017, pp. 1–3.
[18] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 3074–3082.
[19] Y. Qi *et al.*, "Hedged deep tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4303–4311.
[20] 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. Workshops*, Dec. 2015, pp. 58–66.
[21] Y. Song, C. Ma, L. Gong, J. Zhang, R. Lau, and M.-H. Yang. (2017). "CREST: Convolutional residual learning for visual tracking." [Online]. Available: https://arxiv.org/abs/1708.00225

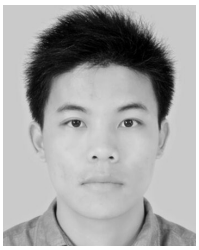
[22] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr. (2017). "End-to-end representation learning for correlation filter based tracking." [Online]. Available: https://arxiv.org/abs/1704.06036
[23] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. (2017). "DCFNet: Discriminant correlation filters network for visual tracking." [Online]. Available: https://arxiv.org/abs/1704.04057
[24] H. Fan and H. Ling. (2017). "Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking." [Online]. Available: https://arxiv.org/abs/1708.00153
[25] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 3119–3127.
[26] L. Wang, W. Ouyang, X. Wang, and H. Lu, "STCT: Sequentially training convolutional networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1373–1381.
[27] 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, May 2017.
[28] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4293–4302.
[29] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 749–765.
[30] 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.* New York, NY, USA: Springer, 2016, pp. 850–865.
[31] D. Gordon, A. Farhadi, and D. Fox. (2017). "Re3: Real-time recurrent regression networks for object tracking." [Online]. Available: https://arxiv.org/abs/1705.06368
[32] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1420–1429.
[33] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.
[34] A. Blake and M. Isard, "The condensation algorithm-conditional density propagation and applications to visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 361–367.
[35] S. Kastner and L. G. Ungerleider, "Mechanisms of visual attention in the human cortex," *Annu. Rev. Neurosci.*, vol. 23, no. 1, pp. 315–341, 2000.
[36] B. Cheung, E. Weiss, and B. Olshausen. (2016). "Emergence of foveal image sampling from learning to attend in visual scenes." [Online]. Available: https://arxiv.org/abs/1611.09430
[37] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 1125–1134.
[38] Z. Chen, Z. Hong, and D. Tao. (2015). "An experimental survey on correlation filter-based tracking." [Online]. Available: https://arxiv.org/abs/1509.05520
[39] 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.*, Dec. 2015, pp. 4310–4318.
[40] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1430–1438.
[41] J. Ning, J. Yang, S. Jiang, L. Zhang, and M.-H. Yang, "Object tracking via dual linear structured SVM and explicit feature map," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4266–4274.
[42] Y. Sui, Z. Zhang, G. Wang, Y. Tang, and L. Zhang, "Real-time visual tracking: Promoting the robustness of correlation filter learning," in *Proc. Eur. Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 662–678.
[43] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 1135–1143.
[44] 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.* New York, NY, USA: Springer, 2016, pp. 472–488.
[45] 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.*, Oct. 2017, pp. 1–9.
[46] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. Cambridge, MA, USA: MIT Press, 1998.
[47] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. (2017). "A brief survey of deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1708.05866
[48] Y. Zhu *et al.* (2017). "Visual semantic planning using deep successor representations." [Online]. Available: https://arxiv.org/abs/1705.08080
[49] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2488–2496.
[50] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[51] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[52] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[53] J. Choi, J. Kwon, and K. M. Lee. (2017). "Visual tracking by reinforced decision making." [Online]. Available: https://arxiv.org/abs/1702.06291
[54] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang. (2017). "Deep reinforcement learning for visual object tracking in videos." [Online]. Available: https://arxiv.org/abs/1701.08936
[55] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 105–114.
[56] J. Supancic, III, and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 322–331.
[57] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1349–1358.
[58] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

Bineng Zhong received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 2004, 2006, and 2010, respectively. From 2007 to 2008, he was a Research Fellow with the Institute of Automation and the Institute of Computing Technology, Chinese Academy of Sciences. From 2017 to 2018, he was a Visiting Scholar with Northeastern University, Boston, MA, USA. He is currently a Professor with the School of Computer Science and Technology, Huaqiao University, Xiamen, China. His current research interests include pattern recognition, machine learning, and computer vision.

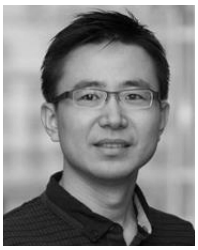
Bing Bai received the B.S. degree in computer science from the Xiamen University of Technology, Xiamen, China, in 2015. He is currently pursuing the master's degree with the School of Computer Science and Technology, Huaqiao University, Xiamen. His current research interests include pattern recognition, machine learning, and computer vision.



Jun Li received the B.A. degree in applied mathematics from Panzhihua University in 2006, the M.S. degree in computer application from China West Normal University in 2009, and the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology in 2015. From 2012 to 2013, he was a Visiting Student with the Department of Statistics, Rutgers University, Piscataway, NJ, USA. He is currently a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. His current research interests include deep learning, sparse representations, subspace clustering, and recurrent neural networks.



Yulun Zhang received the B.E. degree from the School of Electronic Engineering, Xidian University, China, in 2013, and the M.E. degree from the Department of Automation, Tsinghua University, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northeastern University, USA. His research interests include image restoration and deep learning. He was a recipient of the Best Student Paper Award at the IEEE International Conference on Visual Communication and Image Processing in 2015.



Yun Fu (S'07–M'08–SM'11–F'19) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xi'an Jiaotong University, China, and the M.S. degree in statistics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign. He has been an Interdisciplinary Faculty Member with the College of Engineering and the College of Computer and Information Science, Northeastern University, since 2012. He has authored or co-authored many papers in leading journals, books/book chapters, and international conferences/workshops. His research interests are machine learning, computational intelligence, big data mining, computer vision, pattern recognition, and cyber-physical systems. He serves as an Associate Editor, the chair, a PC member, and a Reviewer for many top journals and international conferences/workshops. He is a fellow of IAPR , OSA, and SPIE, a Lifetime Distinguished Member of ACM, a Lifetime Member of AAAI and Institute of Mathematical Statistics, a member of the ACM Future of Computing Academy, Global Young Academy, AAAS, and INNS, and a Beckman Graduate Fellow in 2007 and 2008. He received seven prestigious young investigator awards from NAE, ONR, ARO, IEEE, INNS, UIUC, and Grainger Foundation; nine Best Paper Awards from IEEE, IAPR, SPIE, and SIAM; and many major industrial research awards from Google, Samsung, and Adobe. He is currently an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEANING SYSTEMS.