Robust Target Tracking by Online Random Forests and Superpixels

Wei Wang, Chunping Wang, Si Liu, *Member, IEEE*, Tianzhu Zhang, *Member, IEEE*, and Xiaochun Cao, *Senior Member, IEEE*

***Abstract*— This paper presents a robust joint discriminative appearance model-based tracking method using online random forests and mid-level feature (superpixels). To achieve superpixel-wise discriminative ability, we propose a joint appearance model that consists of two random forest-based models, *i.e.*, the background-target discriminative model (BTM) and the distractor-target discriminative model (DTM). More specifically, the BTM effectively learns discriminative information between the target object and the background. In contrast, the DTM is used to suppress distracting superpixels, which significantly improves the tracker's robustness and alleviates the drifting problem. A novel online random forest regression algorithm is proposed to build the two models. The BTM and DTM are linearly combined into a joint model to compute a confidence map. Tracking results are estimated using the confidence map, in which the position and scale of the target are estimated orderly. Furthermore, we design a model updating strategy to adapt the appearance changes over time by discarding degraded trees of the BTM and DTM and initializing new trees as replacements. We test the proposed tracking method on two large tracking benchmarks, the CVPR2013 tracking benchmark and VOT2014 tracking challenge. Experimental results show that the tracker runs at real-time speed and achieves favorable tracking performance compared with the state-of-the-art methods. The results also suggest that the DTM improves tracking performance significantly and plays an important role in robust tracking.**

***Index Terms*— Vision tracking, online random forests, joint discriminative model, superpixels.**

I. Introduction

VISION tracking is a fundamental and active research topic in computer vision, which has a wide range of applications, such as autonomous cars, surveillance, human computer interaction, biomedical image analysis, and the military. Given an object's initial state (indicating the position and scale of the target object) in the first frame of an image sequence, the task of tracking is to estimate the state of the object in the remaining frames [1]. It remains very attractive to design a robust tracker which can handle various challenging situations [2], [3], such as illumination change, geometric deformation, fast motion, occlusion, and background clutter.

Numerous outstanding algorithms have been proposed for robust object tracking in recent years that focus on constructing an effective appearance model by capturing low-level vision cues or high-level structural cues. Low-level cues are obtained by various descriptors or operators such as color, gradient, texture, local histogram or local binary pattern. While the design of low-level features is elaborate [7], they are not effective at tracking because of their limited performance when representing targets [8]. For example, Haar-like features are frequently used by tracking methods [4], [5], [9] for their superior ability to handle severe appearance and illumination change. However, these tracking methods are less effective at handling scale change and occlusion. High-level structural cues usually convey semantic or prior knowledge about a target object, such as target geometry or contour. For example, [10] exploited an object silhouette to track non-rigid target objects in occlusion situations. However, to obtain preferable performance, their method needs to know an object's shape priors which are not available when tracking an arbitrary object. In brief, tracking with high-level structural cues is limited to specific situation or object because extracting these cues for arbitrary object remains an unsolved problem [11].

Taking into account the limitations of low-level vision cues and high-level structural cues in tracking, mid-level visual cues are suitable for tracking because they are more effective than low-level cues at representing objects and less complex than high-level cues. As a mid-level cue, superpixels group pixels into perceptually meaningful atomic regions [12] that provides a natural and convenient representation of an image and greatly reduce the complexity of subsequent processing tasks. In recent years, superpixels have gained much attention in some computer vision tasks, such as semantic segmentation [13] and object localization [14]. Several superpixel based tracking methods have been proposed [15], [16]. Their tracking results demonstrate the effectiveness of superpixel representation at handling challenging situations.

In this paper, we present a target tracking method that constructs two superpixel-based appearance models. Basically, we train a Background-Target discriminative Model (BTM) to distinguish target object from its surroundings. However,

Manuscript received October 8, 2015; revised April 1, 2016 and June 6, 2016; accepted July 7, 2016. Date of publication March 20, 2017; date of current version July 2, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61572493 and Grant U1536203, in part by the 100 Talents Programme of The Chinese Academy of Sciences, and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA06010701. This paper was recommended by Associate Editor J.-M. Odobez. *(Corresponding author: Xiaochun Cao.)*

W. Wang is with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with the Second Department, Ordnance Engineering College, Shijiazhuang 050003, China (e-mail: wang_wei.buaa@163.com).

C. Wang is with the Second Department, Ordnance Engineering College, Shijiazhuang 050003, China (e-mail: wchp@tom.com).

S. Liu and X. Cao are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: liusi@iie.ac.cn; caoxiaochun@iie.ac.cn).

T. Zhang is with the State Key Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100090, China (e-mail: tzzhang10@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2017.2684759

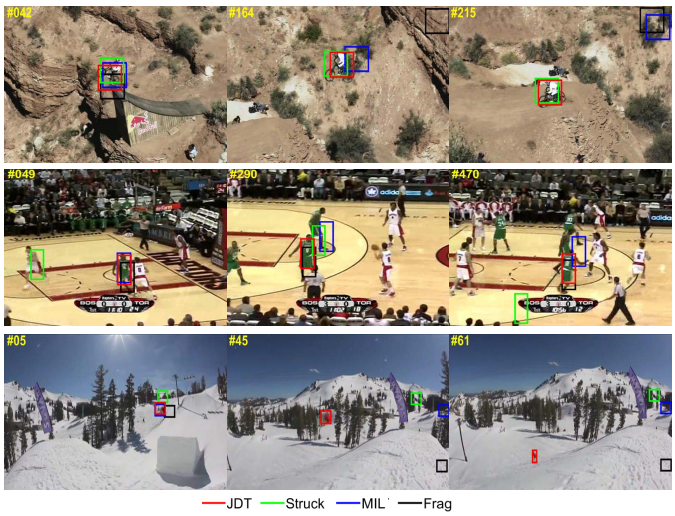


Fig. 1. Comparison of the proposed JDT and Struck [4], MIL [5], and Frag [6] in challenging situations such as deformation, distracting object, and in-plane rotation. The example frames are from the *MountainBike*, *Basketball*, and *Skiing* sequences, respectively. The images are best viewed in color.

a tracker with only this single model tends to drift towards regions with an appearance that is similar to the target. To suppress these regions explicitly, we train an additional specific model called Distractor-Target discriminative Model (DTM) that significantly alleviates the risk of drifting. The BTM and DTM are trained by a proposed online random forest regression algorithm and combined into a joint discriminative appearance model. We refer to the proposed tracking method as the Joint Discriminative appearance model based Tracking method (JDT). Using online random forests and superpixels, the proposed JDT achieves real-time tracking speed and better performance than state-of-the-art methods. Some tracking results are shown in Fig. 1.

A. Overview of the Proposed Tracker

An overview of the proposed method is shown in Fig. 2. In the t-th frame, tracking is divided into two stages, training and estimating.

1) Training: In this stage, the target state in the current frame has been obtained at the estimating stage or given manually (only in the 1st frame). We train two discriminative appearance models based on the target state. First, we crop a target object and its surrounding region and segment the region into superpixels, as illustrated in Fig. 2(d). We refer to the superpixels in the surrounding (blue region) as background superpixels, and the others as target superpixels. We then label these superpixels as background or target. The BTM is then trained (or updated) using the background and target superpixels.

Next, we crop a new target/surrounding region that is bigger than the above-mentioned region. We segment the new region into superpixels and use the trained BTM to evaluate them (see Fig. 2(d-e)). As illustrated in Fig. 2(e), some superpixels, which do not belong to the target but exhibit a similar appearance to the target, are given high confidences by the BTM. We refer to these superpixels as distracting superpixels or distractors. Distractors may degrade the tracking method. The DTM is trained (or updated) using the distractors and target superpixels. We illustrate the concepts of distractors, background and target superpixels in Fig. 2 in a more intuitive manner.

To train the BTM and DTM in online mode, an online random forest regression algorithm is proposed in this paper. Here, the BTM and DTM are two random forests that are linearly combined into a joint discriminative appearance model. Furthermore, a model updating mechanism is designed to adapt to the appearance changes of the target and background. The updated BTM and DTM are passed on to the next frame.

2) Estimating: During the estimating stage, a search region is extracted from the incoming frame and segmented into superpixels. Each of these superpixels is jointly scored by the trained BTM and DTM. A confidence map is then generated to indicate the probability of each pixel belonging to the target. According to the confidence map, we first estimate the target position quickly using an integral image method. The target scale is then estimated based on the estimated position. The estimated state is used to update the BTM and DTM at the current frame and then passed on to the next frame.

B. Contributions

The main contributions of this paper are as follows.

First, we propose an online random forest regressor for robust superpixel-based target tracking. To the best of our knowledge, this is the *first* online random forest algorithm for the regression task that has the same properties as traditional random forests, *i.e.*, high computational efficiency while preserving discriminative power. Because of the proposed online random forest regressor, our tracker achieves the desired speed and precision.

Second, we construct a joint discriminative appearance model that consists of two models, *i.e.* the BTM and DTM, by explicitly taking into account the background and distractors, respectively. The DTM is used to suppress distractors which significantly improves the tracker's robustness and alleviates the drift problem.

The rest of the paper is organized as follows. Related work is reviewed in Section II. An online random forest regressor is proposed in Section III. Our tracking method is further discussed in Section IV. Section V reports and discusses the experimental results, which show the effectiveness of the proposed tracker. Section VI provides the concluding remarks.

II. Related Work

A. Vision Tracking

We focus on model-free short-term online tracking in this paper, where model-free means the only supervised training example is provided by the bounding box in the first frame [17]. This task has been extensively studied over the past few decades. We refer readers to two recent surveys [1], [18] for more details. In general, most of the proposed tracking methods in recent years are tracking-by-detection methods. They pose tracking as a detection problem

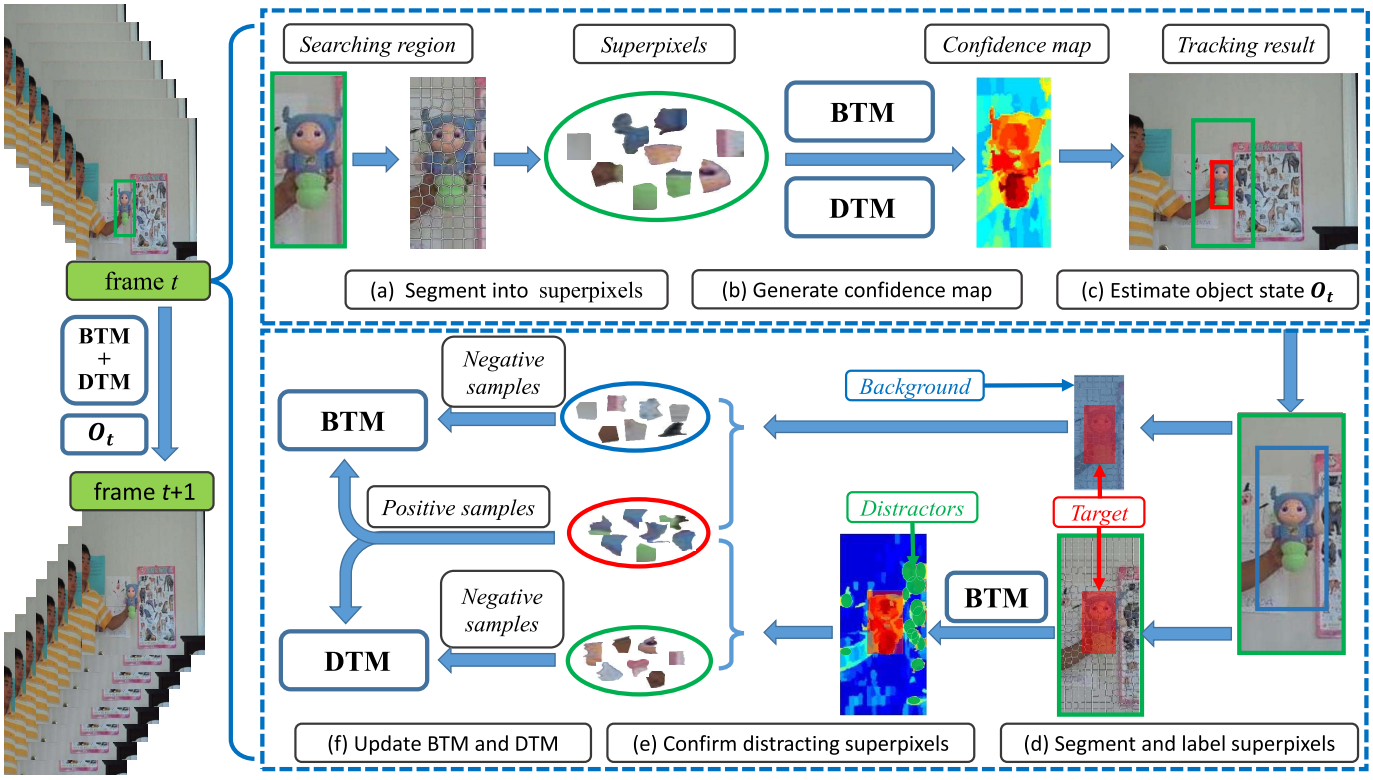


Fig. 2. Overview of the proposed JDT method. The two dashed boxes represent the two stages of the JDT, *i.e.* estimating target state and training the joint model, respectively. In the estimating stage, a confidence map is generated to estimate the target state. In the training stage, we first determine whether the target is occluded. If the target is not occluded, the BTM and DTM are updated. The BTM is designed to distinguish the target object from its background. The DTM is designed to suppress distractors and alleviate the risk of drifting. The positive training data of the BTM and DTM are target superpixels (red region). The negative training data of the BTM and DTM are background superpixels (blue region) and distractors (green region), respectively. Note that all superpixels in the blue region are background superpixels, and only the superpixels (green region) that have similar appearance to the target are labeled as distractors. Finally, the estimated target state and the updated (or not updated, if the target is occluded) BTM and DTM are passed on to the next frame.

and fall into two categories according to the statistical models they use: generative trackers and discriminative trackers.

Generative trackers typically build an appearance model by learning from the target object in the previous frames and the tracking result of the next frame is the candidate that is the most consistent with the appearance model. Several generative tracking methods are originally, namely the IVT [19], the VTD [20] and ℓ_1 tracker [21]. Many other trackers have been inspired by these solutions [22]–[26]. [19] proposed an incremental learning tracking method that built an online subspace to adapt to appearance changes. Kwon and Lee [20] used multiple basic models for combining observation and motion models as well as trackers. The resulting VTD method can track a target object whose motion and appearance changes drastically and combinatorially. In the ℓ_1 tracker [21] and its, the authors built a target and trivial templates set to represent the target object, and formulated tracking as a ℓ_1-regularized least square problem to handle the corrupted appearance. However, most generative trackers tend to drift away from the target in complicated backgrounds [27].

Unlike generative trackers, the appearance models of discriminative trackers are learned from both the target object and its surrounding. The tracking target is separated from a surrounding region by the learned models. [28] presented an online feature selection mechanism that demonstrated the effectiveness of exploiting discriminative features to improve tracking performance. The ensemble tracker proposed in [29] assembled numerous weak classifier into a strong classifier that determines whether pixels belong to the target object. [9] presented an on-line AdaBoost algorithm for real-time object tracking in which the surrounding background was regarded as negative examples. Therefore, this method can choose the most discriminative features between the object and the background. Tracking has inherent ambiguities that are caused by noisy labels. This makes it difficult for traditional supervised learning methods and may consequently cause tracking drift or failure. To avoid this problem, two notable tracking methods [4], [5] have been presented. MILTrack [5] was designed to prevent drift caused by ambiguities in the sample labels, in which online Multiple Instance Learning (MIL) was used to training the appearance model. [4] framed the overall tracking problem as one of structured output prediction using a kernelized structured output Support Vector Machine (SVM) to avoid the labeling ambiguity when updating the classifier during tracking. Recently, correlation filters have widely been used in tracking methods for their high speed [1], [30]–[32]. Heriques *et al.* [30] proposed a fast CSK tracker based on correlation filters and kernel ridge regression. Further, [33] and [34] improved the CSK by introducing multi-channel features and structural information, respectively. They all achieve admirable performance.

They further improved the CSK to a multi-channel feature that achieves admirable performance [33].

Our method is also formulated within the later category. Inspired by [35], in addition to building a discriminative appearance model as in the above discriminative trackers, we also build an extra model to discriminate the target and distractors. The main difference between our method and [35] is that distractors in [35] mean distracting objects, while in our method, distractors mean distracting superpixels which make our method more precise.

B. Superpixel Tracking

Several image segmentation algorithms can produce superpixels. [36] proposed a graph-based segmentation algorithm in which pixel nodes on a graph are clustered. In [37], the mean-shift method was used to obtain a better segmentation iteratively starting from an initial rough clustering. The above two segmentation algorithms have been used for object recognition [38] and image labeling [39]. However, they do not offer an explicit control on the number and size of superpixels, which are important for designing superpixel representations and object model learning. [12] introduced the Simple Linear Iterative Clustering (SLIC) method, which clusters pixels in a combined five-dimensional space to efficiently generate compact, nearly uniform superpixels and has a low computational cost. Recently, SLIC has been used in visual tracking for its effective object representation performance [15]. Therefore, we employ the SLIC method [12] to segment image regions into superpixels.

In [15], a robust tracker named SPT was proposed. SPT incorporated superpixel representations in a discriminative appearance model using the mean shift cluster method. While SPT was proved to be able to recover from drift and demonstrated robustness in their project, it was still unstable when similar distractors appear in the background. [8] applied superpixels to depth video tracking via depth fusion. [40] used superpixels to reduce the sample set in a local-global tracking methodology. More recently, [16] presented a tracking method by designing an adaptive clustered decision tree. In their method, targets were modeled at three levels of granularity and superpixels were used to build a middle level model.

Of these methods, our method is most similar to SPT. However, two differences between our method and SPT should be noted. First, we build the target appearance model using online random forest instead of mean shift cluster, which makes SPT considerably slower. Second, to avoid the situation in which SPT is vulnerable to distractors, we construct a joint discriminative appearance model. These improvements make our tracker faster and more robust.

C. Online Learning

To adapt to changes of the target and surroundings, appearance models must be updated online in tracking. For example, online MIL, online Adaboost, and online SVR were proposed in [5], [9], [41], respectively. Random forests achieve competitive predictive performance [42] and are computationally efficient to train and test. They can also be parallelized easily, making them suitable for online tracking. Most existing online random forests either need to store previous data (and are not real online methods) or have a predictive performance that is inferior to their offline versions [43]. Only the Mondrian forests in [43] and online forests in [44] have achieved predictive performance that is competitive with offline random forests while being faster. [43] employed Mondrian processes to construct ensembles of random decision trees. In [44], an online random forest was proposed by combining an on-line bagging and extremely randomized forests method. However, the above two online forests are both designed for the classification task, and they cannot be directly used for our tracking method, in which an online regression algorithm is needed. Hence, we learn from [44] and propose an online random forest regression algorithm in Section III.

III. Online Random Forest Regressor

Our tracking method is based on random forests (RFs), which has shown excellent performance in traditional machine learning tasks [42]. A random forest is an ensemble of randomized decision trees. Denote a random forest as $F = \{f_m\}_{m=1}^{M}$, where f_m is the m-th tree of the random forest, and M is the number of trees in the forest. Each tree in F is trained and tested independently from the other trees. In a trained random forest F, each node in tree f_m is a learned split. Given a test sample x, each tree decides x to left or right down the tree using trained splits until a leaf node is reached. Then tree t_m gives x a prediction $f_m(x)$ according to the reached leaf node state. The output of forest F on x is obtained by combining the predictions of all trees using an ensemble model. We use a common ensemble model in this paper, *i.e.*, an averaging model, as follows,

$$F(x) = \frac{1}{M} \sum_{m} f_m(x). \tag{1}$$

Conventional random forests are trained in offline mode, *i.e.* training data is given in advance. However, training data arrives sequentially in tracking, and training and testing are performed alternately. Moreover, a regression model is needed in our situation. Therefore, we propose an online random forest regressor in Section III-A.

A. Online Training of the Decision Tree

The outstanding learning capacity of a random forest comes from the high variance between decision trees. Two processes are needed to ensure the high variance between trees: i) the training data of tree is randomly sampled and ii) a node's split function is generated randomly. The online random forest proposed in [44] has the desired properties i) and ii). First, each tree f_m is trained on each sample d times, where d is a random number generated by a Poisson(λ) distribution. This ensures each tree will have a different training set. Parameter λ of the Poisson distribution is set to one in the experiments. Hence, some samples will be ignored with about a 30% probability during training a tree. Second, each tree is an extremely randomized tree in which the split functions and thresholds

Algorithm 1 Online Random Forest Regressor

Initialization: Size of the forest: M; minimum number of samples: n_{split}; the minimum gain: δ_{split}.
Input: Random Forest; sample (x, y).
01. **for** $m = 1$ to M **do** //update each tree
02. $d =$ Possion(1)
03. **Repeat** d times:
04. spread x to a leaf node j of tree m
05. update $\{n, \mu_n, E_n\}_j$ for leaf node j and $\{n, \mu_n, E_n\}_{j,p,l}$ or $\{n, \mu_n, E_n\}_{j,p,r}$ for each split using Eq. 5-9
06. **if** $|X_j| > n_{split}$ and $\exists$ split *s.t.* $\Delta E_{j,p} > \delta_{split}$ **then**
07. split node j, use the split $(g_p(x), \theta_p) = \arg\max_p(\Delta E_{j,p})$
08. create left and right children as new leaf node.
09. **end if**
10. **end Repeat**
11. **end for**
Output: Random Forest.

are all generated randomly [45]. We inherit the two processes from [44].

The training of a tree starts from its root node, and several candidate splits $\{(g_p(x), \theta_p)\}_{p=1}^{P}$ are generated randomly in the node, where $g_p(x)$ is a split function that maps sample x to a real value, and θ_p is a scalar threshold. Each split determines a propagating side for each test sample. The training goal is to choose a 'good' split according to some quality measurements. Specifically, a split function randomly chooses two dimensions in the feature space of samples k_1 and k_2. Then,

$$g_p(x) = x^{k_1} \cdot w_1 + x^{k_2} \cdot w_2 \tag{2}$$

where $\omega_{k,j} = rand[-1, 1]$. If $g_p(x) > \theta_s$, the split sends x to the left otherwise, it is sent to the right.

For node j, we use a common quality measurement, the gain function, to determine the good split. The gain function is defined as follows,

$$\Delta E_{j,p} = E(X_j) - \frac{|X_{j,p,l}|}{|X_j|} \cdot E(X_{j,p,l}) - \frac{|X_{j,p,r}|}{|X_j|} \cdot E(X_{j,p,r}) \tag{3}$$

where $E(\cdot)$ denotes the Shannon entropy, X_j denotes the training samples arriving at node j, $X_{j,p,l}$ and $X_{j,p,r}$ are the left and right partitions made by split $(g_p(x), \theta_p)$ and $|\cdot|$ denotes the number of samples in it. Note that $\Delta E_{j,s} \geq 0$.

In this paper, we formalize tracking as a regression task using online random forests. For single-variate regression, Shannon entropy is defined as follows [46],

$$E(X) = \frac{1}{|X|} \sum_i (y_i - \mu)^2 \tag{4}$$

where $\mu = \frac{1}{|X|} \sum_i (y_i)$ is the average value of labels in X.

Training samples arrive one-by-one in tracking, and the tree must be updated as soon as data arrive. We maintain a triple $\{n, \mu_n, E_n\}$ for each leaf node and its partitions to compute the entropy of data set X that is already assigned to the node, where $n = |X|$. When a new sample $\{x_{n+1}, y_{n+1}\}$ arrives, the average value and Shannon entropy of a leaf node or a partition are updated as follows,

$$\mu_{n+1} = (n \cdot \mu_n + y_{n+1})/(n + 1) \tag{5}$$

$$E_{n+1} = \frac{1}{n+1} \sum_{i=1}^{n+1} (y_i - \mu_{n+1})^2 \tag{6}$$

where the sum item in Eq. 6 is rewritten as follows,

$$\sum_{i=1}^{n+1} (y_i - \mu_{n+1})^2 = \sum_{i=1}^{n} (y_i - \mu_{n+1})^2 + (y_{n+1} - \mu_{n+1})^2 \tag{7}$$

$$(y_i - \mu_{n+1})^2 = (y_i - \mu_n)^2 - 2\triangle\mu \cdot (y_i - \mu_n) + (\triangle\mu)^2 \tag{8}$$

where $\triangle\mu = \mu_{n+1} - \mu_n$. We obtain,

$$E_{n+1} = \frac{1}{n+1}\{\sum_{i=1}^{n} (y_i - \mu_n)^2 + 2\triangle\mu \cdot \sum_{i=1}^{n} (y_i - \mu_n) + n \cdot (\triangle\mu)^2 + (y_{n+1} - \mu_{n+1})^2\} \tag{9}$$

the first and second items in the brace are equal to nE_n and zero, respectively. We employ Eq. 5 and Eq. 9 to update the node triples and their splits states.

Following Eq. 5-9, we maintain a triple set that consists of $\{n, \mu_n, E_n\}_j$ for leaf node j, and several $\{n, \mu_n, E_n\}_{j,p,l}$ and $\{n, \mu_n, E_n\}_{j,p,r}$ for each candidate split of node j. Gain function $\Delta E_{j,p}$ is conveniently computed. We then introduce two parameters: 1) n_{split}, which is the minimum number of samples a leaf node has to see before splitting, and 2) δ_{split}, which is the minimum gain a split has to achieve before splitting, as [44]. When $|X_j| > n_{split}$ and existing split $(g_p(x), \theta_p)$ satisfies $\Delta E_{j,p} > \delta_{split}$, the leaf node j is splited using $(g_p(x), \theta_p)$.

After a split occurs, the split leaf node becomes a parent node and two child nodes are created as new leaves. We propagate the average values $\mu_{j,p,l}$ and $\mu_{j,p,r}$ of the good split to the newly created left and right leaf nodes, respectively. Therefore, a new leaf node can give a sample arriving at this node prediction $\mu_{j,p,l}$ or $\mu_{j,p,r}$.

The proposed online random forest regressor is detailed in Algorithm 1.

IV. Proposed Method

We present our tracking method in this section. First, the superpixel-based joint discriminative appearance model, including the BTM and DTM, is constructed using an online random forest. We then compute a confidence map and efficiently estimate the object based on this map. The model update scheme is presented at the end of this section.

A. Joint Discriminative Appearance Model

In this section, we construct a superpixel-based joint discriminative appearance model to distinguish an object from the surrounding background and distractors using the proposed online random forest. The state of the target object at frame t is obtained at the estimating stage of frame t, denoted as $O_t = \{u_t, v_t, h_t, w_t\}$, where $\{u_t, v_t\}$ represents the center coordinate of the target, and h_t and w_t represent the width and height of the target, respectively. O_t is also used to represent the circumscribed bounding box of the target object.

1) Superpixel Representation: The SLIC method [12] is employed to segment image region into superpixels. The parameters of SLIC are constant in each tracking stage, as described in Section V-A. The number of superpixel produced by SLIC is similar to the expected number. It has been proved that color [35], [47] and HOG features [33], [48] help a tracker achieve satisfactory performance. Therefore, we extract color and HOG feature to represent superpixels. The color name space is an effective way to represent pure color information, and empirically shows more discriminative ability in distinguishing pixels than other color spaces [47]. Therefore, we map pixels from RGB space to the color attributes space [49] and normalize them to a probabilistic 10-dimensional color representation space, which is the same as used in [47]. A superpixel's color feature is defined as the average and standard deviation of the color features of its pixels (*i.e.* the pixels within the superpixel). For the HOG feature, we exact HOG descriptors with a cell size of 4 pixels within the circumscribed rectangle of a superpixel. The superpixel's HOG feature is defined as the average of the HOG descriptors of these cells. A superpixel is represented as a 51-dimensional vector (20 for the color attribute space and 31 for the HOG descriptor), and is denoted by ftr.

2) Background-Target Discriminative Model: We employ the proposed online random forest regressor to learn the superpixel-based BTM, which is a random forest of M_1 trees. At each frame, the BTM is trained (updated) to distinguish target object superpixels from its surrounding background superpixels.

Assume pixels within O_t belong to the target, and vice-versa. A pixel in frame t is labeled as follows,

$$y(pixel(u,v;t)) = \begin{cases} \phi((u,v), O_t) & \text{if } pixel(u,v;t) \in O_t \\ 0 & \text{if } pixel(u,v;t) \in B_t \end{cases} \tag{10}$$

where (u, v) is the coordinate of the pixel, $B_t = R_t \setminus O_t$ is the background surrounding region(see Fig. 2(d)), and R_t is a rectangle region centered at (u_t, v_t) of height and width equal to $\alpha_1 \cdot (h_t, w_t)$, where $\alpha_1 > 1$ is a constant parameter. In addition, $\phi(\cdot)$ is a score function based on the ℓ_2 distance to evaluate the probability that a pixel belongs to the target,

$$\phi((u,v), O_t) = \frac{1}{2} + \frac{1}{2}\exp\left(-\frac{\|(u,v) - (u_t, v_t)\|^2}{(h_t/2)^2 + (w_t/2)^2}\right) \tag{11}$$

where $\phi((u, v), O_t) \in [0.68, 1]$ for $pixel(u, v; t) \in O_t$. Hence, the closer the pixel is to the center $\{(u_t, v_t)\}$, the higher its probability of belonging to the target.

We segment R_t into superpixels and extract the features of these superpixels. Each superpixel is labeled according to its pixels,

$$y(sp(n;t)) = \text{mean}(\{y(pixel(u,v;t)) | pixel(u,v;t) \in sp(n)\}) \tag{12}$$

where mean$(\cdot)$ computes average of the elements of the input set, n is a superpixel's index. For conciseness, we denote $y(sp(n; t))$ as $y_t(n)$. The labels of the background superpixels are then equal to 0, and the labels of the target superpixels are nonzero numbers.

It is not clear how to judge whether a superpixel belongs to O_t or B_t, because some superpixels intersect both O_t and B_t. Fortunately, our RF is a regressor that only needs the label value of each superpixel and does not need to judge whether a superpixel belongs to O_t or B_t. The proposed RF regressor applies the training data $\{ftr_t(n), y_t(n)\}_n$ one-by-one to train the BTM.

3) Distractor-Target Discriminative Model: The trained BTM is used to work out a confidence map for a rectangle region based on the target state at the current frame. How the confidence map is obtained is discussed in Section IV-B.1. The BTM learns to distinguish the target from the background; however, we observe that some non-target superpixels (*i.e.* distractors) have high confidence (see Fig. 2(e)). The tracker is susceptible to distractors and further drift. Updating the BTM using the distractors as weighted negative samples can weaken the problem of drift. Instead, we learn an additional DTM to overcome this issue. We do this because the latter approach suppresses the distractors in a more explicit and meaningful manner. Furthermore, if we use the samples repeatedly to increase their weights in the BTM, it may increase the computational cost.

In practice, we crop a rectangle $\hat{R}_t$, called training region, from the current frame. Rectangles $\hat{R}_t$ and R_t are concentric: the parameters of $\hat{R}_t$ are $\{u_t, v_t, \alpha_2 h_t, \alpha_2 w_t\}$, where $\alpha_1 < \alpha_2$. We calculate a confidence map for $\hat{R}_t$ using the BTM and compute mean confidence score C_m for pixels in O_t. The distracting superpixels, which are in $\hat{R}_t \setminus O_t$ and have higher confidence score than $0.7C_m$, are then found out. These distractors are referred to as D_t. The distractors and target superpixels are labeled the same as in Eq. 10-12. These labeled superpixels are used by the proposed online RF method to train the DTM, which is an RF with M_2 trees. The BTM and DTM are then linearly combined into a joint discriminative appearance model.

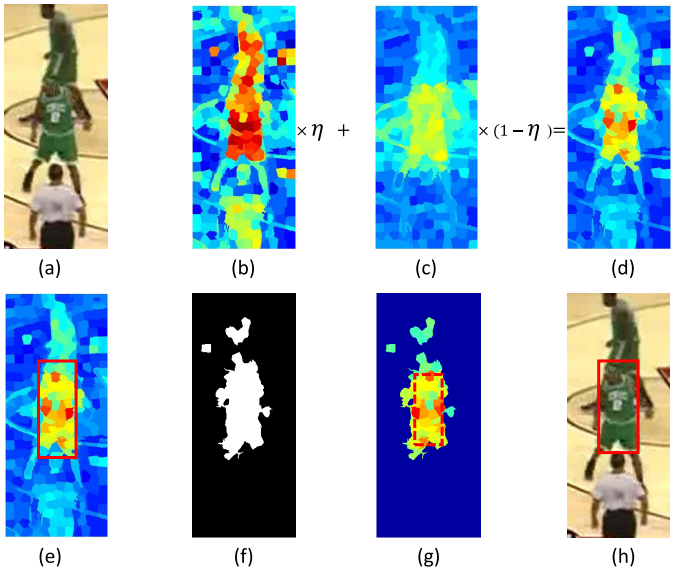


Fig. 3. **Illustration of target state estimation.** (a) Searching region $\bar{R}_{t+1}$. (b) BTM confidence map. (c) DTM confidence map. (d) Combined confidence map using Eq. 14. (e) Estimated target position with the target scale of the previous frame. (f) Segmentation of the confidence map by threshold τ^*. (g) Connected components and safe foreground region. (h) Final tracking results for this frame. Confidence maps are shown using a thermodynamic diagram, where hot colors correspond to high confidence value.

B. Estimating the Target State

When frame t arrives, we use the BTM and DTM learnt in the training stage of frame $t-1$ to compute a confidence map. The new object state is estimated based on this confidence map.

1) Confidence Map Computation: At first, we extract a searching region $\bar{R}_t$ from the incoming t-th frame. The parameters of $\bar{R}_t$ are $\{u_{t-1}, v_{t-1}, \alpha_3 h_{t-1}, \alpha_3 w_{t-1}\}$, where α_3 controls the size of the search region. Then, $\bar{R}_t$ is segmented into N_t superpixels using the SLIC method. For each superpixel, confidence scores $C_{t,B}(n)$ and $C_{t,D}(n)$ are evaluated by the BTM and DTM, respectively, using Eq. 1. Then $C_{t,B}(n)$ and $C_{t,D}(n)$ are combined to form a joint confidence score using the linear interpolation,

$$C_t(n) = \eta \cdot C_{t,B}(n) + (1-\eta) \cdot C_{t,D}(n) \tag{13}$$

where η is the interpolation coefficient.

Next, we assign the combined score $C_t(n)$ of superpixel $sp(n;t)$ to its pixels and obtain a confidence map that indicates the probability that the pixel belongs to the target, as illustrated in Fig. 3(d).

2) Target State Estimation: This confidence map is sufficient for most real applications [50]. However, tracking requires the state of the object as its final output, so we need to estimate the best bounding box based on the confidence map. Similar to recent scale-adaptive trackers such as [35], [48] our detection strategy is to first localize the target position and then estimate the target size.

To localize the target position, we densely sample overlapping target candidates within the search region $\bar{R}_t$, using fixed target size (h_{t-1}, w_{t-1}). The candidate with the maximum confidence is determined to be the object,

$$O_t^* = \arg\max_{O_t^l}(C(O_t^l)) \tag{14}$$

where O_t^l is the l-th candidate state, $C(O_t^l)$ is the confidence score of the state,

$$C(O_t^l) = \sum_{pixel(u,v)\in O_t^l} C_t(pixel(u,v)) \tag{15}$$

Score C_t^l in Eq. 15 is computed using an integral image method, making the target position localize very fast.

To estimate the target size, we choose to extend the scale estimation algorithm in [35] because of its availability and effectiveness. Given target position $O_t^* = \{u_t^*, v_t^*, h_{t-1}, w_{t-1}\}$, threshold τ^* is computed to segment the confidence map,

$$\tau^* = \arg\max_{\tau\in\{0.1,\ldots,0.9\}} (r(O_t^*, \tau) - r(\bar{R}_t \setminus O_t^*, \tau)) \tag{16}$$

where $r(O_t^*, \tau)$ represents the ratio of pixels whose confidence score exceeds τ in O_t, and $r(\bar{R}_t \setminus O_t^*, \tau)$ is similar. In Eq. 16, τ^* is a trade-off that be set to retain as many pixels as possible in O_t^* while removing as many pixels as possible in $\bar{R}_{t+1} \setminus O_t^*$.

The confidence map is then segmented by τ^* (see Fig. 3(f)), and a connected component analysis is performed on the segmentation result (see Fig. 3(d)). Next, a safe foreground region, found by shrinking O_t^* to its inner 80%, is asserted to belong to the target. The connected components that intersect with the safe foreground region are also asserted to belong to the target. Final object state O_t is determined by computing the enclosing bounding box over these regions (see Fig. 3(g)).

C. Model Updating and Occlusion Handling

1) Model Updating: Our joint discriminative appearance model is updated in two hierarchies to adapt and capture the continuous appearance changes of the target and its background. In the basic hierarchy, each tree of the BTM and DTM is updated using the proposed online random forest regressor. While this takes the new appearance of the target into account, the old appearance from previous frames with large intervals is preserved, which may be useless and sometimes even harmful to the current tracking. To avoid this, we also update the BTM and DTM in a higher hierarchy, which allows the unlearning of old information.

In practice, we discard partial trees of the BTM and DTM and consequently initialize new trees to replace the discarded trees. In the training stage, the superpixels in the training region are labeled by Eq. 12. We use the BTM and DTM to estimate the test errors of each tree. We then discard the top-z trees and initialize z new trees to replace these discarded trees. The discarding is performed independently for the BTM and DTM. Because the impact of a few trees on an ensemble of trees is relatively low, discarding them usually does not harm the performance of the entire forest [44].

Algorithm 2 The Proposed JDT Method

Initialization: Initial state of the target object: O_1; frame number of sequence: T; initial occlusion factor: $Occ_1 = 1$; initial occlusion threshold: $\theta_1 = 0.8$.

01. **for** $t = 1$ to T **do**
    Estimating
02. **if** $t > 1$
03. Crop a search region from the t-th frame according O_{t-1} and compute a confidence map. (Section IV-B.1)
04. Localize the target position using Eq. 14. (Section IV-B.2)
05. Estimate target state O_t based on the estimated target position. (Section IV-B.2)
06. Compute occlusion factor Occ_t and threshold θ_t. (Section IV-C)
07. **end if**
    Training/Updating
08. **if** $Occ_t > \theta_t$
09. Discard and initialize τ trees according to the test errors of the trees in both the BTM and DTM. (Section IV-C)
10. Train the BTM by Algorithm 1, using target and background superpixels as training data. (Section IV-A.2)
11. Train the DTM by Algorithm 1, using target and distracting superpixels as training data. (Section IV-A.3)
12. **end if**
13. **end for**

2) Occlusion Handling: Updating the model may degrade our tracking method when the target is under heavy occlusion. Therefore, we design an occlusion detection mechanism and only update the target model when occlusion is not detected. We compute the mean confidence for target O_t as occlusion factor

$$Occ_t = \frac{1}{h_t \cdot w_t} C(O_t). \tag{17}$$

An occlusion threshold θ_t is also defined, where $\theta_t = 0.8\hat{\theta}_t$ and $\hat{\theta}_t$ is the average of the mean confidence in the previous H frames. If the occlusion factor Occ_t is less than θ_t, this indicates that there are many background pixels in the bounding box of the estimated target and it is very likely to be occluded. When the target is considered to be occluded, we do not update the BTM and DTM.

The main steps of our tracking system are summarized in Algorithm 2.

V. Experiments

In this section we evaluate the proposed JDT tracking method by comparing it with state-of-the-art methods on a large publicly available benchmark dataset, namely the CVPR2013 online tracking benchmark [1]. The benchmark contains 50 sequences and covers almost all the challenging factors in tracking, including non-rigid deformation, pose and scale variation, occlusion, *etc*. Moreover, our JDT is tested on the Visual Object Tracking challenge 2014 datasets (VOT14) [17].

A. Experiment Setup

1) Implementation Details: We assume the movement of target and its background is smooth, *i.e.* the location of the target at the next frame is equally likely to appear within a radius of the tracker location in the current frame. In addition, we must suppress distractors in advance by using a large region to identify them. Hence, we use the expanding parameters $\alpha_1 = 1.8$, $\alpha_2 = 3$, and $\alpha_3 = 2$, which correspond to the three regions used to train the BTM, as well as the DTM and estimate the target state. The three regions are resized to be uniform and the target boxes are fixed in size at 32×32 pixels. According to the experiment set in [15], the spatial proximity weight parameters in the SLIC method are set to 8 for all three regions because their sizes are uniform, and the numbers of superpixels in SLIC are set to 100, 170, and 110 for the three regions, respectively. By resizing the three regions and setting suitable parameters for the SLIC method, we obtain approximately consistent superpixel segment results, which is important for our tracker. We give equal importance to the BTM and DTM in our method by setting the interpolation coefficient η to 0.5 and the numbers of trees in the BTM/DTM to $M_1 = M_2 = 20$. The number 20 is a trade-off between the speed and performance of the proposed tracker (we offer more detailed experimental results in Section V-B). The number z of discarded trees in the BTM/DTM is set to one tenth of M_1/M_2, *i.e.* 2. Parameters P (Section III-A) and H (Section IV-C) are set to 20 and 10, respectively.

Because each tree in a forest is built and tested independently of the other trees, our tracker is performed in parallel during our experiment. Implemented in MATLAB, our tracker runs at about 23 frames per second (fps) on an i7 3 GHz machine with eight workers.

2) Evaluation Criteria: For the CVPR2013 benchmark, we follow the one-pass evaluation criteria of the CVPR2013 benchmark, *i.e.* running a tracker throughout a test sequence with initialization from the groundtruth in the first frame. The tracking performance is measured with respect to two aspects, distance precision and overlap success rate. Distance precision is represented by a precision plot, which shows the percentage of frames whose estimated location is within the given threshold distance of the groundtruth. The overlap success rate is represented by success plot, which reveals the percentage of frames where the overlap rate between the bounding boxes of the groundtruth and estimated target state

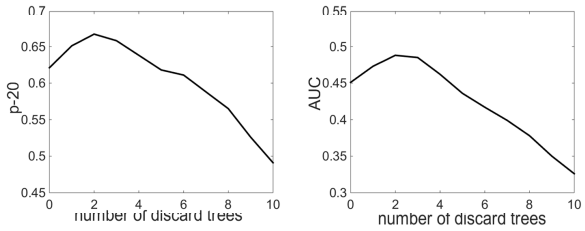


Fig. 4. Sensitivity analysis of the JDT to the number of discarded trees z.

TABLE I
CONFIGURATION AND TRACKING RESULTS OF SEVERAL VARIETIES OF OUR JDT

Methods	M_1	M_2	tpf/ms	p-20	AUC
JDT10	10	10	25.5	0.491	0.326
JDT15	15	15	35.1	0.614	0.436
JDT20	20	20	44.2	0.667	0.488
JDT25	25	25	56.3	0.679	0.495
JDT30	30	30	67.2	0.685	0.501
BDT	40	0	43.9	0.588	0.392

surpasses a threshold. We also use the percentage score in the threshold = 20 pixels and the area under curve (AUC) of the success plot as the representative precision and success score [1], [5], respectively.

For the VOT14 benchmark, trackers are initialized in the first frame using the groundtruth and re-initialized after tracking failure. The VOT14 framework provides a ranking analysis based on two metrics, accuracy (average bounding box overlap) and robustness (number of re-initializations). We follow the rank analysis, and refer readers to [17] for a detailed description of this metric.

B. Analysis of Proposed Method

The sensitivity of the JDT to parameters z, the number of discarded trees, is illustrated in Fig. 4. The p-20 and AUC metrics are representative values of the overall overlap success and distance precision on the whole CVPR2013 benchmark for different value of z. This figure shows that the best tracking performance is achieved when z is equal to 2, as we set in Section V-A.

We then implement five more varieties of our tracking method by setting different numbers of trees in the BTM and DTM. JDT denotes the default setting of our method, *i.e.* M_1 and M_2 are both set to 20. First, we implement four varieties by setting $M_1 = M_2 = 10$, $M_1 = M_2 = 15$, $M_1 = M_2 = 25$ and $M_1 = M_2 = 30$, denoted as JDT10, JDT15, JDT25, and JDT30, respectively. These comparison experiments provide a guide to the choice of M_1 and M_2. Table I shows the setting and representative performance of the varieties of JDT. In Table I, we can see that the tracking time consumed per frame (tpf) is approximately proportional to $M_1 + M_2$, but the tracking performance does not linearly increase. Increasing $M_1 + M_2$ from 20 to 40, *i.e.*, from JDT10 to JDT, p-20 and AUC improve significantly, by about 18% and 16%, respectively, and the tracking speed is halved. Increasing $M_1 + M_2$ from 40 to 60, *i.e.*, from JDT to JDT30, p-20 and AUC improve only by about 2% and 1%, respectively, and the tracking speed is again reduced by half. Setting $M_1 = M_2 = 20$, *i.e.* JDT, is a trade-off between speed and performance, where we achieve both competitive performance and real-time speed.

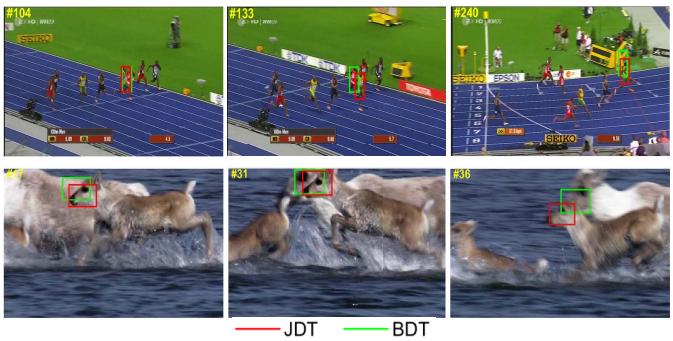


Fig. 5. Screenshots of JDT and BDT tracking with the *Bolt* and *Deer* sequences to demonstrate the effectiveness of the proposed joint model. First row: *Bolt* sequence and second row: *Deer* sequence.

We also implemented a variety of JDT by setting $M_1 = 40, M_2 = 0$, denoted as BDT. The BDT only uses the BTM and has the same number of trees with JDT. It can be seen in Table I, using the joint DTM and BTM model improves the performance significantly by about 8% and 10%, respectively, with respect to p-20 and AUC. Furthermore, we use an example to demonstrate the effectiveness of the proposed joint model in Fig. 5. In the *Bolt* sequence, the target bounding boxes estimated by the JDT and BDT all contain a portion of a distracting runner at frame 104. In this case, the BDT loses the target (Bolt) and drifts to the distracting runner at frame 133, because the distracting runner confuses and degrades the BTM after frame 104. In contrast, the JDT performs well at tracking the target because it uses the proposed DTM, which suppresses the distractor and alleviates the risk of drifting. Because the BDT has drifted, we re-initialize it using the groundtruth annotation at frame 133 to further compare the JDT and BDT. Note that we do this re-initializing operation on the CVPR2013 benchmark only in this example and only for BDT. At frame 240, there are a large number of distracting superpixels in the surrounding region. Unfortunately, the BDT is misled by these distracting superpixels and loses the target again. In contrast, the JDT is able to track the target object in this challenging situation because the DTM is employed to search for these distracting superpixels in advance (remember, we use large expanding parameters for region $\bar{R}$), and the DTM then learns to suppress these distracting superpixels. In the second row of Fig. 5, the tracking target is the head of a little deer. Its body and the other deer have a similar appearance to the head. The BDT drifts several times, but JDT is still able to track the target. These experimental results prove that BTM is not sufficient for tracking the target object under challenging situations and the DTM is an important component of a robust tracking method.

C. Experimental Results on CVPR2013 Benchmark

For a fair evaluation, the proposed JDT was evaluated on the CVPR2013 Benchmark and compared against nine

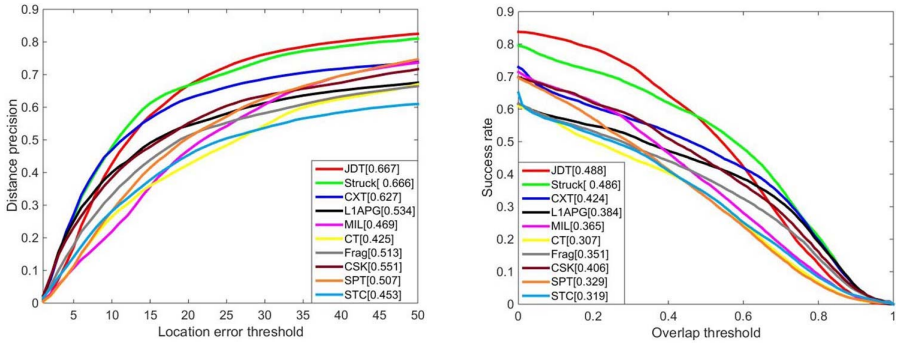


Fig. 6. Distance precision and overlap success rate over 50 CVPR benchmark sequences. The legend contains the representative precision and success scores for each tracker. The proposed JDT method performs better than other trackers.

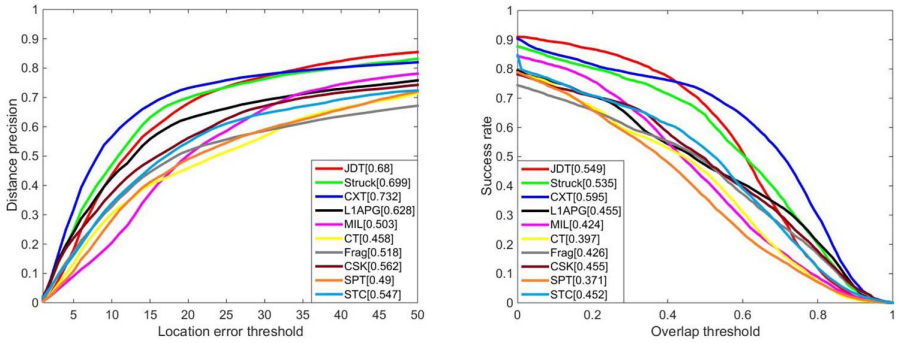


Fig. 7. Precision plots and success plots for 31 sequences in which the target object rotates in the image plane.

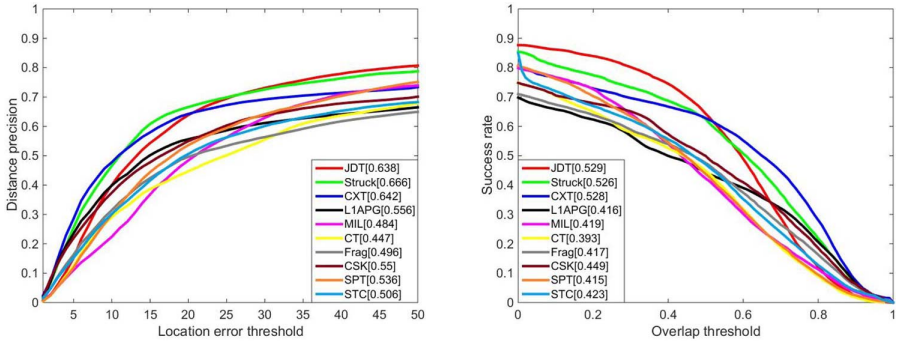


Fig. 8. Precision plots and success plots for 39 sequences in which the target object rotates out the image plane.

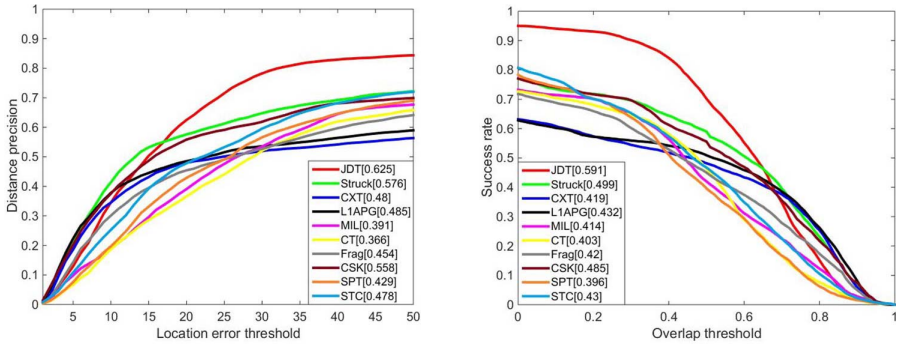


Fig. 9. Precision plots and success plots for 19 sequences in which the non-rigid object deformation.

state-of-the-art tracking methods, including two of the top-performing trackers of the benchmark, namely Struck [4], and CXT [51]. Furthermore, we use a sparse representation-based L1APG [52] with a holistic intensity template, part-based tracker Frag [6] with color histograms, discriminative Tracker MIL [5], and CT [53] with a holistic Haar-like template, and two correlation trackers (CSK [30] and STC [31]) with holistic intensity or color templates. The tracking results or source code of these comparisons were provided by the benchmark or corresponding authors. We also evaluated the SPT [15]

Fig. 10. Screenshots of JDT on three attributes sequences of the CVPR2013 benchmark, compared with nine other state-of-the-arts tracking methods. (Best viewed in color). These screenshots were taken at the midpoints of the 42 videos with non-rigid object deformation, in-plane rotation, or out-of-plane rotation in the benchmark [1], [5]. As the three sets have several same sequences, the total number of sequences belong to the three attributes sets is 43. (The *Bolt* sequence is shown in Fig. 5.)

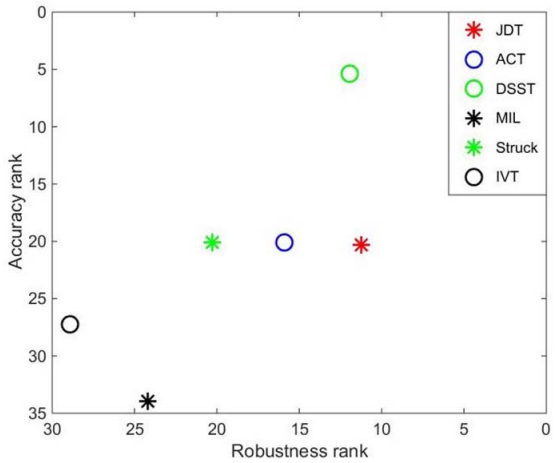


Fig. 11. Ranking results for the VOT14 benchmark dataset. Top-performing trackers are located top-right.

method, which uses superpixels and has a similar framework to our JDT method. As the SPT method needs to set unique dynamic distribution parameters for each target object before tracking, which is not fair to other trackers, we used the SPT fixed parameters that are provided in their code.

As shown in Fig. 6, our tracker achieves comparable overall performance with Struck [4] and CXT [51], and performs favorably against the other trackers both in distance precision and overlap success rate. Particularly, compared with the SPT [15] method, the proposed JDT has a significant improvement of about 16% for p-20 and 22% for AUC. This indicates that the random forest used in the JDT is more effective at exploiting discriminative information of superpixels than the mean shift clustering used in SPT. In Fig. 6, we can see that our tracker shows a high success rate for low overlap, and its performance drops significantly in high precision areas. This may be because we do not design a customized feature for superpixels, and the tracking bounding box is not as tight as the best Struck tracker. When the overlap threshold is 0.5, which is often used for object detection or tracking evaluation, our tracker achieves results that are comparable to the best tracker Struck.

Furthermore, our tracker significantly outperforms other methods on the sequences with three attributes, *i.e.*, non-rigid object deformation, in-plane rotation, and out-of-plane rotation. We report the results for these three attributes in Fig. 7-9. The JDT method achieves competitive results on sequences with target in-plane rotation and out-of-plane rotation, as well as Struck [4] and CXT [51] methods, as shown in Fig. 7 and 8. We attribute these results to the adaptive update of our joint model, in which two online random forests are updated by discarding old trees and initializing new trees at each frame. In terms of non-rigid object deformation,

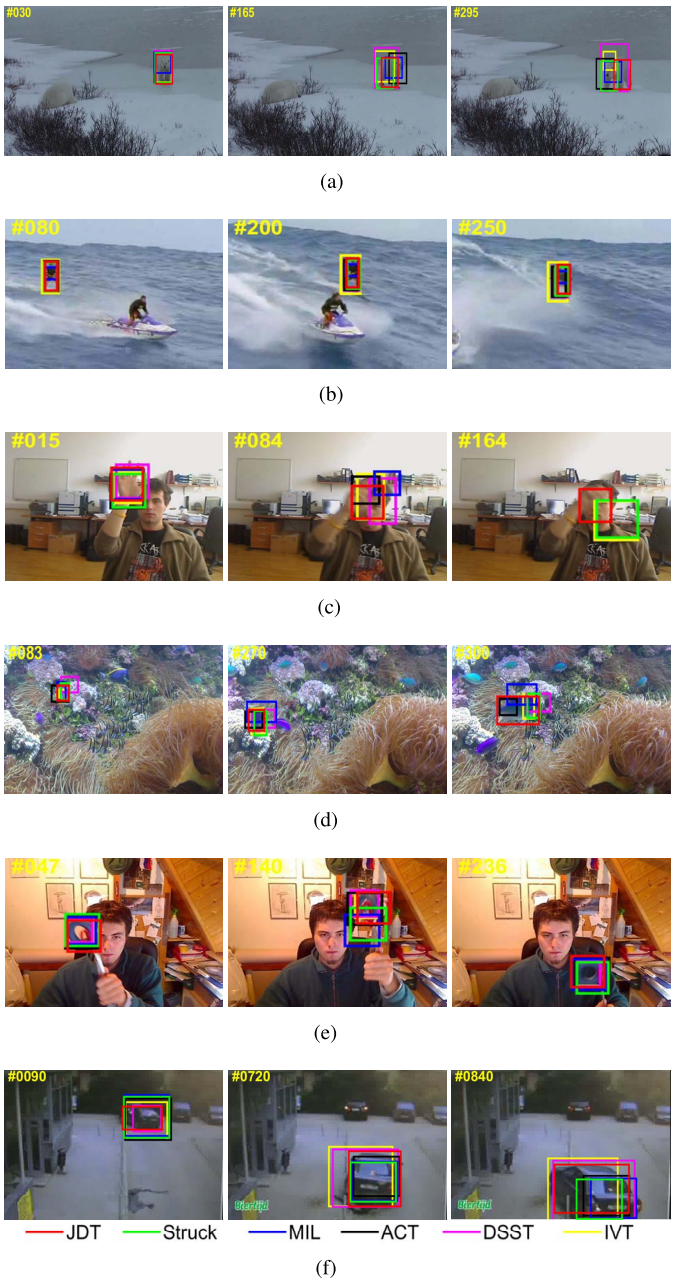


Fig. 12. Screenshots of JDT tracking on several challenging sequences of VOT14, compared with five other state-of-the-arts tracking methods. Failing trackers are not shown in these screenshots. (a) *polarbear* sequence. (b) *surfing* sequence. (c) *hand2* sequence. (d) *fish1* sequence. (e) *torus* sequence. (f) *drunk* sequence.

which is more challenging than the other two attributes, the performance of most tracking methods, including the proposed JDT, drops significantly. However, the JDT still performs better, with value of p-20 and AUC up to 62.5% and 59.1%, respectively. This is 4.9% and 9.2% better than the p-20 and AUC of the Struck method, as shown in Fig. 9. We attribute the strong performance under non-rigid object deformation to the effective representation of our joint discriminative appearance model.

In Fig. 10, we present screenshots of the 42 videos with the above attributes. We report these tracking results by taking the screenshots at the midpoint of the sequences. This is fairer than choosing a special frame for each sequence. We can see the JDT successfully deals with most sequences in Fig. 10.

D. Experimental Results on VOT14

Additionally, we evaluate our JDT method on the VOT14 benchmark dataset [17] and compare it against six state-of-the-art tracking methods, including Struck [4], MIL [5], a color-based tracker ACT [47], the winner of the VOT14 challenge DSST [48], and incremental learning tracking method IVT [19]. There are 25 sequences (some sequences are also selected in the CVPR2013 benchmark) in VOT14. The overall comparison results on VOT14 are shown in Fig. 11 in terms of accuracy rank and robustness rank. As can be seen from the ranking results in Fig. 11, our method outperforms all competitors with respect to robustness rank and also achieves comparable performance with ACT [47] and Struck [4] with respect accuracy rank.

Tracking results for several challenging sequences of VOT14 are shown in Fig. 12. In the *polarbear* and *surfing* sequences, the targets have a similar appearance to the background. The proposed JDT method performs well because it is designed to discriminate the target from the background using the BTM. In contrast, IVT [19], as a generative tracker, only models the target appearance, which makes its tracking results tend to drift (see frame 295 of *polarbear*) and contain too much of the background region (see frame 200 and 250 of *surfing*). In Fig. 12(c), the target is a swinging hand, which can be distracted by the forearm and face. All other trackers have drifted to the forearm or face or lost the target. In contrast, our JDT builds the DTM to distinguish target superpixels from distracting superpixels and tracks the hand accurately throughout the *hand2* sequence. In the *fish1*, *torus* and *drunk* sequences in Fig. 12, the proposed JDT is able to keep track of the targets, even when these targets undergo non-rigid deformation, rotation, and large scale changes. Without an adaptive scale estimation strategy, Struck [4] and MIL [5] do not accurately locate targets. While ACT [47], DSST [48], and IVT [19] perform well when tracking rigid targets, such as the car (*drunk* sequence) and torus (*torus* sequence), they fail to estimate the scale of the target in *fish1* sequence, which undergoes non-rigid shape deformation and heavy out-of-plane rotation.

VI. Conclusion

In this paper, we present a real-time robust tracking method called JDT that demonstrates the effectiveness of superpixel representation in tracking. First, an online random forest regression algorithm is proposed to build a joint discriminative appearance model, which consists of the BTM and DTM. In the joint model, we explicitly take into account both the background and distracting superpixels, therefore yielding a robust target representation. Next, a confidence map is computed based on the BTM and DTM to estimate the target state. Furthermore, we design a model updating strategy to adapt to appearance changes over time. Extensive experiments show that the proposed JDT tracker has favorable tracking

performance compared to many state-of-the-art trackers in the presence of various challenges, especially deformation and rotation.

REFERENCES

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

[2] T. A. Biresaw, A. Cavallaro, and C. S. Regazzoni, "Tracker-level fusion for robust Bayesian visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 5, pp. 776–789, May 2015.

[3] Y. Wu, B. Shen, and H. Ling, "Visual tracking via online nonnegative matrix factorization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 3, pp. 374–383, Mar. 2014.

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

[5] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[6] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, Jun. 2006, pp. 798–805.

[7] X. Wen, L. Shao, Y. Xue, and W. Fang, "A rapid learning algorithm for vehicle classification," *Inf. Sci.*, vol. 295, pp. 395–406, Feb. 2015.

[8] Y. Yuan, J. Fang, and Q. Wang, "Robust superpixel tracking via depth fusion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 15–26, Jan. 2014.

[9] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. Brit. Mach. Vis. Conf.*, Edinburgh, U.K., Sep. 2006, pp. 47–56.

[10] A. Yilmaz, X. Li, and M. Shan, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1531–1536, Nov. 2004.

[11] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 1619–1632, 2006.

[12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[13] J. Wu, Y. Zhao, J.-Y. Zhu, S. Luo, and Z. Tu, "MILCut: A sweeping line multiple instance learning paradigm for interactive image segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 256–263.

[14] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proc. IEEE Int. Conf. Comput. Vis.*, Kyoto, Japan, Sep./Oct. 2009, pp. 670–677.

[15] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 1323–1330.

[16] J. Xiao, R. Stolkin, and A. Leonardis, "Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 4978–4987.

[17] M. Kristan *et al.*, "The visual object tracking VOT2014 challenge results," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland, Sep. 2014, pp. 191–217.

[18] 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.

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

[20] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 1269–1276.

[21] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Kyoto, Japan, Sep./Oct. 2009, pp. 1436–1443.

[22] T. Zhang *et al.*, "Structural sparse tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 150–158.

[23] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Object tracking with joint optimization of representation and classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 4, pp. 638–650, Apr. 2015.

[24] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem, "Robust visual tracking via consistent low-rank sparse learning," *Int. J. Comput. Vis.*, vol. 111, no. 2, pp. 171–190, 2015.

[25] L. Wang, H. Yan, K. Lv, and C. Pan, "Visual tracking via kernel sparse representation with multikernel fusion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 7, pp. 1132–1141, Jul. 2014.

[26] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via structured multi-task sparse learning," *Int. J. Comput. Vis.*, vol. 101, no. 2, pp. 376–383, 2013.

[27] N. Wang, J. Shi, D. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, Dec. 2015, pp. 3101–3109.

[28] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1631–1643, Oct. 2005.

[29] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[30] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, Oct. 2012, pp. 702–715.

[31] 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.*, Zurich, Switzerland, Sep. 2014, pp. 127–141.

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

[33] 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.

[34] L. Si, T. Zhang, X. Cao, and C. Xu, "Structural correlation filter for robust visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun./Jul. 2016, pp. 4312–4320.

[35] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 2113–2120.

[36] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[37] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[38] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.

[39] Q. Zhou, J. Zhu, and W. Liu, "Learning dynamic hybrid Markov random field for image labeling," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2219–2232, Jun. 2013.

[40] L. Čehovin, M. Kristan, and A. Leonardis, "Superpixel segmentation for robust visual tracking," in *Proc. IEEE Int. Electrotech. Comput. Sci. Conf.*, Portorož, Slovenia, Sep. 2013, pp. 1–4.

[41] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for ν-support vector regression," *Neural Netw.*, vol. 67, pp. 140–150, Jul. 2015.

[42] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[43] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian forests: Efficient online random forests," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 3140–3148.

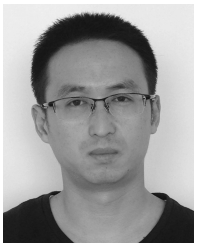
[44] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *Proc. IEEE Int. Conf. Comput. Vis.*, Kyoto, Japan, Sep./Oct. 2009, pp. 1393–1400.

[45] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[46] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Found. Trends Comput. Graph. Vis.*, vol. 7, nos. 2–3, pp. 81–227, Feb. 2012.

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

[48] 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., Sep. 2014, pp. 1–11.
[49] J. van de Weijer, C. Schmid, J. Verbeek, and D. Larlus, "Learning color names for real-world applications," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1512–1523, Jul. 2009.
[50] N. Wang, S. Li, A. Gupta, and D. Yeung. (2015). "Transferring rich feature hierarchies for robust visual tracking." [Online]. Available: http://arxiv.org/abs/1501.04587
[51] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Colorado Springs, CO, USA, Jun. 2011, pp. 1177–1184.
[52] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust ℓ_1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 1830–1837.
[53] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. Eur. Conf. Comput. Vis.*, Florence, Italy, Oct. 2012, pp. 864–877.



Wei Wang received the B.E. degree in automation from Beihang University, Beijing, China, in 2010 and the M.E. degree from the Ordnance Engineering College, Shijiazhuang, China, in 2013, where he is currently working toward the Ph.D. degree. In 2015, he was an Intern Student with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing. His research interests include object tracking and object detection.

Chunping Wang received the M.E. degree in signal and information processing from Hebei University of Technology, Tianjin, China, in 1991 and the Ph.D. degree from the Ordnance Engineering College, Shijiazhuang, China, in 1999.

He is currently a Professor with the Ordnance Engineering College. His research interests include object tracking, object recognition, and fire control theory and system.

Si Liu received the bachelor's degree from the Experimental Class, Beijing Institute of Technology, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences. She was a Research Fellow with the Learning and Vision Research Group, Department of Electrical and Computer Engineering, National University of Singapore. She is currently an Associate Professor with the Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include object categorization, object detection, image parsing, and human pose estimation.

Tianzhu Zhang received the B.S. degree in communications and information technology from Beijing Institute of Technology in 2006 and the Ph.D. degree in pattern recognition and intelligent systems from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2011. He was with the Advanced Digital Sciences Center of Singapore. He is currently an Associate Professor with National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests are in computer vision and multimedia, including action recognition, object classification, and object tracking.

Xiaochun Cao received the B.E. and M.E. degrees in computer science from Beihang University, Beijing, China, and the Ph.D. degree in computer science from University of Central Florida, Orlando, FL, USA. He spent about three years with ObjectVideo Inc., as a Research Scientist. From 2008 to 2012, he was a Professor with Tianjin University, Tianjin, China. He has been a Professor with Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, since 2012. He has authored or co-authored over 120 journal and conference papers.

Dr. Cao is a Fellow of the IET. He is on the Editorial Board of IEEE TRANSACTIONS OF IMAGE PROCESSING. His dissertation was nominated for the University of Central Floridas university-level Outstanding Dissertation Award. In 2004 and 2010, he was a recipient of the Piero Zamperoni Best Student Paper Award at the International Conference on Pattern Recognition.