DeepTrack: Learning Discriminative Feature Representations Online for Robust Visual Tracking

Hanxi Li, Yi Li, and Fatih Porikli, *Fellow, IEEE*

***Abstract*— Deep neural networks, albeit their great success on feature learning in various computer vision tasks, are usually considered as impractical for online visual tracking, because they require very long training time and a large number of training samples. In this paper, we present an efficient and very robust tracking algorithm using a single convolutional neural network (CNN) for learning effective feature representations of the target object in a purely online manner. Our contributions are multifold. First, we introduce a novel truncated structural loss function that maintains as many training samples as possible and reduces the risk of tracking error accumulation. Second, we enhance the ordinary stochastic gradient descent approach in CNN training with a robust sample selection mechanism. The sampling mechanism randomly generates positive and negative samples from different temporal distributions, which are generated by taking the temporal relations and label noise into account. Finally, a lazy yet effective updating scheme is designed for CNN training. Equipped with this novel updating algorithm, the CNN model is robust to some long-existing difficulties in visual tracking, such as occlusion or incorrect detections, without loss of the effective adaption for significant appearance changes. In the experiment, our CNN tracker outperforms all compared state-of-the-art methods on two recently proposed benchmarks, which in total involve over 60 video sequences. The remarkable performance improvement over the existing trackers illustrates the superiority of the feature representations, which are learned purely online via the proposed deep learning framework.**

***Index Terms*— Convolutional neural network, deep learning, visual tracking.**

Manuscript received March 25, 2015; revised September 22, 2015 and November 23, 2015; accepted December 3, 2015. Date of publication December 22, 2015; date of current version March 9, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61462042 and Grant 61503168, and in part by the Australian Research Council's Discovery Projects Funding Scheme under Project DP150104645. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ling Shao. *(Corresponding author: Yi Li.)*

H. Li is with the School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, Jiangxi 330022, China, and also with the Canberra Research Lab, National ICT Australia, ACT 2601, Australia (e-mail: hanxi.li@nicta.com.au).

Y. Li is with the Toyota Research Institute, North America, Ann Arbor, MI 48105 USA, also with the Canberra Research Lab, National ICT Australia, ACT 2601, Australia, and also with the Research School of Information Science and Engineering, Australian National University, Acton, ACT 2600, Australia (e-mail: liyi.umd@gmail.com).

F. Porikli is with Canberra Research Lab, National ICT Australia, ACT 2601, Australia, and also with the Research School of Information Science and Engineering, Australian National University, Acton, ACT 2600, Australia (e-mail: fatih.porikli@nicta.com.au).

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/TIP.2015.2510583

I. Introduction

IMAGE features play a crucial role in many challenging computer vision tasks such as object recognition and detection. Unfortunately, in many *online* visual trackers features are manually defined and combined [1]–[7]. Even though these methods report satisfactory results on individual datasets, hand-crafted feature representations would limit the performance of tracking. For instance, normalized cross correlation, which would be discriminative when the lighting condition is favourable, might become ineffective when the object moves under shadow. This necessitates good representation learning mechanisms for visual tracking that are capable of capturing the appearance effectively changes over time.

Recently, deep neural networks have gained significant attention thanks to their success on learning feature representations. Different from the traditional hand-crafted features [8]–[10], a multi-layer neural network architecture can efficiently capture sophisticated hierarchies describing the raw data [11]. In particular, the Convolutional Neural Networks (CNN) has shown superior performance on standard object recognition tasks [12]–[16], which effectively learn complicated mappings while utilizing minimal domain knowledge.

However, the immediate adoption of CNN for online visual tracking is not straightforward. First of all, CNN requires a large number of training samples, which is often not available in visual tracking as there exist only a few number of reliable positive instances extracted from the initial frames. Moreover, CNN tends to easily overfit to the most recent observation, *e.g.*, most recent instance dominating the model, which may result in drift problem. Besides, CNN training is computationally intensive for online visual tracking. Due to these difficulties, CNN has been treated as an offline feature extraction module on predefined datasets [17], [18] for tracking applications so far.

In this work, we propose a novel tracking algorithm using CNN to automatically learn the most useful feature representations of particular target objects while overcoming the above challenges. We employ a tracking-by-detection strategy – a four-layer CNN model to distinguish the target object from its surrounding background. Our CNN generates scores for all possible hypotheses of the object locations (motion states) in a given frame. The hypothesis with the highest score is then selected as the prediction of the motion state in the current frame. We update this CNN model in an purely online manner.

In other words, the proposed tracker is learned based only on the samples obtained from the current video sequence; no extra information or offline training is required.

Typically, tracking-by-detection approaches rely on predefined heuristics to sample from the estimated object location to construct a set of positive and negative samples. Often these samples have binary labels, which leads to a few positive samples and a large negative training set. However, it is well-known that CNN training without any pre-learned model usually requires a large number of training samples, both for positive ones and negative ones. Furthermore, even with sufficient samples, the learner usually needs hundreds of seconds to achieve a CNN model with an acceptable accuracy. The slow updating speed could prevent the CNN model from being a practical visual tracker. To address these two issues, our CNN model employs a special type of loss function that consists of a structural term and a truncated norm. The structural term makes it possible to obtain a large number of training samples that have different significance levels considering the uncertainty of the object location at the same time. The truncated norm is applied on the CNN response to reduce the number of samples in the back-propagation [12], [13] to significantly accelerate the training process.

We employ the Stochastic Gradient Decent (SGD) method to optimize the parameters in the CNN model. Since the standard SGD algorithm is not tailored for online visual tracking, we propose the following two modifications. First, to prevent the CNN model from overfitting to occasionally detected false positive instances, we introduce a *temporal sampling mechanism* to the batch generation in the SGD algorithm. This temporal sampling mechanism assumes that the object patches shall stay longer than those of the background in the memory. Therefore, we store all the observed image patches into training sample pool, and we choose the positive samples from a temporal range much longer than the negative ones. In practice, we found this is a key factor in the robust CNN-based tracker, because discriminative sampling strategy successfully regularizes the training for effective appearance model. Secondly, the object locations, except the one on the first frame, is not always reliable as they are estimated by the visual tracker and the uncertainty is unavoidable [19]. One can treat this difficulty as the label noise problem [20]–[22]. We propose to sample the training data in the joint distribution over three random variables, i.e. the frame index, the image patch index and the label noise state. In the experiment, further performance improvement is observed when the label noise is taken into account.

For achieving a high generalization ability in various image conditions, we use multiple image *cues* (low-level image features, such as normalized gray-scale image and image gradient) as independent channels as network input. We update the CNN parameters by iteratively training each channel independently followed by a joint training on a fusion layer which replaces the last fully-connected layers from multiple channels. The training processes of the independent channels and the fusion layer are totally decoupled. Empirically, we observed that this two-stage iterative procedure is more accurate than jointly training for all cues.

Finally, we propose to update the CNN model in a "lazy" style. First, the CNN-model is only updated when a significant appearance change occurs on the object. The intuition behind this lazy updating strategy is that we assume that the object appearance is more consistent over the video, compared with the background appearances. Second, the fusion layer is updated in a coordinate-descent style and with a lower learning rate. The underlying assumption is that the feature representations can change fast while each image cue contributes to the final classification task in a more stable manner over all the frames. In practice, this lazy updating strategy not only increases the tracking speed significantly but also yields observable accuracy increase.

To summarize, our main contributions include:

- A visual tracker based on online adapting CNN is proposed. As far as we are aware, this is the first time a single CNN is introduced for learning the best features for object tracking in an online manner.
- A structural and truncated loss function is exploited for the online CNN tracker. This enables us to achieve very reliable and robust tracking while achieving tracking speeds up to 4fps.
- An iterative SGD method with an robust temporal sampling mechanism is introduced for competently capturing object appearance changes and meanwhile considering the label noise.

The novel CNN-based tracker, termed *DeepTrack* in this work, outperforms all the compared state-of-the-art algorithms in the experiments on two recently proposed benchmarks involving over 60 videos. In addition, it achieves a practical tracking speed (from 1.5fps to 4fps depending on the sequence and settings), which is comparable to many other visual trackers.

II. CNN Architecture

A. CNN With a Single Image Cue

Our CNN consists of two convolutional layers and two fully-connected layers. The ReLU (Rectified Linear Unit) [13] is adopted as the activation function and max-pooling operators are used for dimension-reduction. The gray and dashed block in Fig. 1 shows the structure of our network, which can be expressed as $(32 \times 32) \rightarrow (10 \times 10 \times 12) \rightarrow (2 \times 2 \times 18) \rightarrow (8) \rightarrow (2)$ in conventional neural network notation.

The input is locally normalized 32×32 image patches, which draws a balance between the representation power and computational load. The first convolution layer contains 12 kernels each of size 13×13 (an empirical trade-off between overfitting due to a very large number of kernels and discrimination power), followed by a pooling operation that reduces the obtained feature map (filter response) to a lower dimension. The second layer contains 216 kernels with size 7×7. This leads to a 72-dimensional feature vector in the second convolutional layer, after the pooling operation in this layer.

The two fully connected layers firstly map the 72-D vector into a 8-D vector and then generate a 2-D confidence vector $\mathbf{s} = [s_1, s_2]^{\mathrm{T}} \in \mathcal{R}^2$, with s_1 and s_2 corresponding to the positive score and negative score, respectively. In order to increase the margin between the scores of the positive and

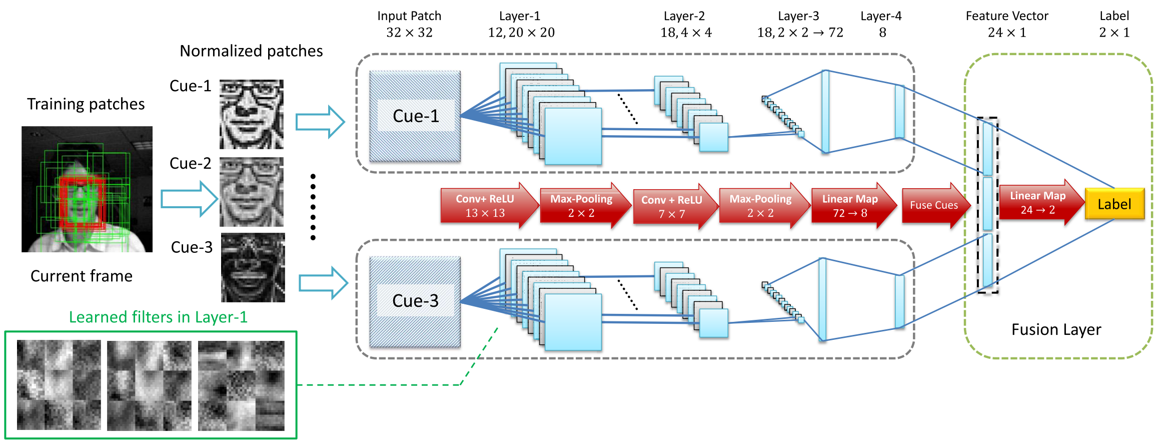


Fig. 1. The architecture of our CNN tracker with multiple image cues. The gray and dashed blocks are the independent CNN channels for different image cues; the green dashed block is the fusion layer where a linear mapping $\mathbb{R}^{24} \rightarrow \mathbb{R}^{2}$ is learned.

negative samples, we calculate the CNN score as

$$S = \text{Score}(\mathbf{s}) = s_1 \cdot \exp(s_1 - s_2), \tag{1}$$

B. CNN With Multiple Image Cues

Effective object tracking requires multiple cues, which may include color, image gradients and different pixel-wise filter responses. These cues are weakly correlated yet contain complementary information. In this work, we conduct the DeepTrack on three image cues. For gray-level images, we employ two local contrast normalized images and one gradient image as the cues. The local contrast normalization, which is robust to illumination changes [13], is used to generate two cues with different parameter configurations. In specific, we use

$$\{r_\mu = 0.1h, r_\sigma = 0.1h\} \text{ and } \{r_\mu = 0.15h, r_\sigma = 0.15h\},$$

where r_μ and r_σ are the two parameters determining the local contrast normalization [13] while h is the height of the current object. For color images, the two locally-normalized cues are simply replaced with the H and V channels of the HSV color representation. Offering multiple image cues, we then let CNN to select the most informative ones in a data driven fashion. By concatenating the final responses of these 3 cues, we build a fusion layer (the green dashed block in Fig. 1) to generate a 2-D output vector, based on which the final CNN score is calculated using Eq. 1.

The proposed DeepTrack algorithm is a patch-based method, which is a common strategy adopted in object detection [23], [24] and visual tracking [4], [25]. However, some recent work on CNN-based detection performs the convolution on the whole image or the ROI (Region of Interest) once and then extracts the score for each image patch in the later stage. This modification stems from the pioneering work [26] decades ago and usually leads to a much efficient detection algorithm [27]–[29] without loss of accuracy. Theoretically, a similar strategy could be used in the CNN-based tracking methods for tracking acceleration while this is beyond the scope of this paper which focus mainly on online feature learning using CNN models.

C. Data Generation and Pre-Processing

In this work, we employ the particle-filter-based motion model [30] which is similar to the one used in [31]. In specific, for each new frame, we sample 1500 random patches in a Gaussian Distribution which centers on the previous predicted state. The new training data is then generated according to the newly estimated motion state and the labeling criteria (Eq. 6) which is explained below. In this work, the standard deviation for the three dimensions are $\min(10, 0.5 \cdot h)$, $\min(10, 0.5 \cdot h)$ and $0.01 \cdot h$, respectively.

Following the common setting in the seminar works [13], [32], we perform the following data pre-processing to achieve a more robust CNN model.

- To better curb the overfitting, all the positive training samples are flipped as augmented data. By assuming that object rarely rotates vertically, we only generate horizontally mirrored samples.
- The pixel values of each the image cue are normalized to the range $[0, 10]$. We found this normalization is crucial for balancing the importances between different image cues.

D. Structural and Truncated Loss Function

1) Structural Loss: Let $\mathbf{x}_n$ and $\mathbf{l}_n \in \{[0,1]^{\mathrm{T}}, [1,0]^{\mathrm{T}}\}$ denote the cue of the input patch and its ground truth label (background or foreground) respectively, and $\mathbf{s} = f(\mathbf{x}_n; \boldsymbol{\Omega})$ be the predicted 2-D vector of $\mathbf{x}_n$ with network weights $\boldsymbol{\Omega}$, the objective function of N samples in the batch is

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \| f(\mathbf{x}_n; \boldsymbol{\Omega}) - \mathbf{l}_n \|_2 \tag{2}$$

when the CNN is trained in the batch-mode. Eq. 2 is a commonly used loss function and performs well in binary classification problems. However, for object localization tasks, usually higher performance can be obtained by 'structurizing' the binary classifier [33], [34]. The advantage of employing the structural loss is the larger number of available training samples, which is crucial to the CNN training. In the ordinary binary-classification setting, one can only use the training samples with high confidences to avoid class ambiguity. In

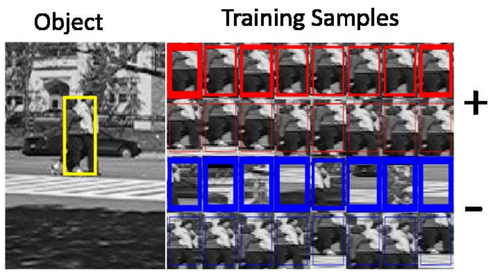


Fig. 2. An illustration of the structural loss defined in Eq. 5 and Eq. 6. The cropped frame (from sequence *couple*) is shown on the left with the predicted region denoted as a yellow bounding box and 4-row of training samples are shown on the right. The upper two rows contain the positive samples (red) while the bottom two store the negative ones (blue). The width of the color boundary indicates the importance $\Delta(\mathbf{y}_n, \mathbf{y}^*)$. We can see some of the instances in row-2 and row-4 are similar. However, their training weights are small so the similar pairs could not influence the training process significantly.

contrast, the structural CNN is learned based upon all the sampled patches.

We modify the original CNN's output to $f(\phi\langle\Gamma, \mathbf{y}_n\rangle; \mathbf{\Omega}) \in \mathbb{R}^2$, where Γ is the current frame, $\mathbf{y}_n \in \mathbb{R}^o$ is the motion parameter vector of the target object, which determines the object's location in Γ and o is the freedom degree[1] of the transformation. The operation $\phi\langle\Gamma, \mathbf{y}_n\rangle$ suffices to crop the features from Γ using the motion state $\mathbf{y}_n$. The associated structural loss is defined as

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left[\Delta(\mathbf{y}_n, \mathbf{y}^*) \cdot \| f(\phi\langle\Gamma, \mathbf{y}_n\rangle; \mathbf{\Omega}) - \mathbf{l}_n\|_2\right], \tag{3}$$

where $\mathbf{y}^*$ is the (estimated) motion state of the target object in the current frame. To define $\Delta(\mathbf{y}_n, \mathbf{y}^*)$ we first calculate the overlapping score $\Theta(\mathbf{y}_n, \mathbf{y}^*)$ defined in [35] as

$$\Theta(\mathbf{y}_n, \mathbf{y}^*) = \frac{\text{area}(r(\mathbf{y}_n) \bigcap r(\mathbf{y}^*))}{\text{area}(r(\mathbf{y}_n) \bigcup r(\mathbf{y}^*))}, \tag{4}$$

where $r(\mathbf{y})$ is the region defined by $\mathbf{y}$, $\bigcap$ and $\bigcup$ denotes the intersection and union operations respectively. Finally we have

$$\Delta(\mathbf{y}_n, \mathbf{y}^*) = \left|\frac{2}{1+\exp(-(\Theta(\mathbf{y}_n, \mathbf{y}^*) - 0.5))} - 1\right| \in [0, 1], \tag{5}$$

and the sample label $\mathbf{l}_n$ is set as

$$\mathbf{l}_n = \begin{cases} [1, 0]^{\mathrm{T}} & \text{if} \quad \Theta(\mathbf{y}_n, \mathbf{y}^*) > 0.5 \\ [0, 1]^{\mathrm{T}} & \text{elsewise} \end{cases} \tag{6}$$

From Eq. 5 we can see that $\Delta(\mathbf{y}_n, \mathbf{y}^*)$ actually measures the importance of the training patch n. For instance, patches that are very close to object center and reasonably far from it may play more significant roles in training the CNN, while the patches in between are less important (see Fig. 2).

In visual tracking, when a new frame $\Gamma_{(t)}$ comes, we predict the object motion state $\mathbf{y}^*_{(t)}$ as

$$\mathbf{y}^*_{(t)} = \arg\max_{\mathbf{y}_n \in \mathcal{Y}} \ \text{Score}\left(f(\phi\langle\Gamma_{(t)}, \mathbf{y}_n\rangle; \mathbf{\Omega})\right), \tag{7}$$

where the score function is defined in Eq. 1 and $\mathcal{Y}$ contains all the test patches in the current frame.

[1]In this paper $o = 3$, *i.e.*, the bounding box changes in its location and the scale.

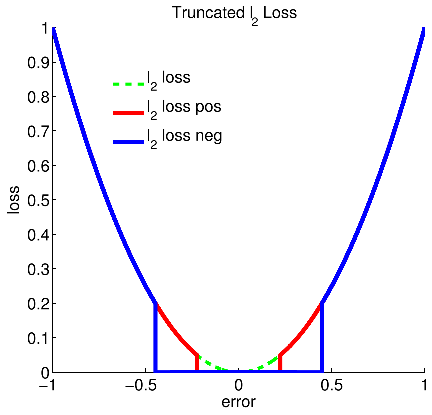


Fig. 3. The truncated l_2 losses. The dashed green curve indicates the original l_2 loss, the red and blue curves are the truncated losses for positive and negative samples.

2) Truncated Structural Loss: Ordinary CNN models regress the input features into the target labels, via the l_2-norm loss. One can directly adopt this strategy in the CNN-based tracking algorithm. However, to speed up the online training process, we employ a truncated l_2-norm in our model. We empirically observe that patches with very small error do not contribute much in the back propagation. Therefore, we can approximate the loss by counting the patches with errors that are larger than a threshold. Motived by this, in [36], we define a truncated l_2 norm as

$$\|e\|_{\mathbb{T}} = \|e\|_2 \cdot \left(1 - [\![\|e\|_2 \le \beta]\!]\right), \tag{8}$$

where $[\![\cdot]\!]$ denotes the indicator function while e is the prediction error, *e.g.*, $f(\phi\langle\Gamma, \mathbf{y}_n\rangle; \mathbf{\Omega}) - \mathbf{l}_n$ for patch-n. In our previous work [36], this truncated loss did increase the training speed, at the cost of slightly reducing the prediction accuracy.

In this work, we observed that the tracking performance is more sensitive to the prediction error on positive samples than the negative samples. Recall that in training stage, we label each positive sample as $[1, 0]^T$ and each negative sample as $[0, 1]^T$. In the test stage, the visual tracker selects the best particle among the ones with high scores. If the highest score in the current frame is large enough, the negative samples with small errors, which are ignored in training according to the truncated loss, will not affect the prediction. In contrast, if one ignores the positive samples with small errors in training, the selection among the top-n particles in the test stage will be consequently inaccurate, and thus drift problems could happen. In other words, we need a more precise loss function for positive samples in visual tracking. We thus improve the original truncated loss function as:

$$\|e\|_{\mathbb{T}} = \|e\|_2 \cdot \left(1 - [\![\|e\|_2 \le \frac{\beta}{(1 + u \cdot l_n)}]\!]\right), \tag{9}$$

where $u > 0$ and $l_n = \mathbf{l}_n(1)$, *i.e.*, the scalar label of the n-th sample. This truncated norm is visualized in Fig. 3 and now Eq. 3 becomes:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\left[\Delta(\mathbf{y}_n, \mathbf{y}^*) \cdot \| f(\phi\langle\Gamma, \mathbf{y}_n\rangle; \mathbf{\Omega}) - \mathbf{l}_n\|_{\mathbb{T}}\right], \tag{10}$$

It is easy to see that with the truncated norm $\|\cdot\|_{\mathbb{T}}$, the backpropagation [12] process only depends on the training samples with large errors, *i.e.*, $\|f(\phi\langle\Gamma, \mathbf{y}_n\rangle; \Omega) - \mathbf{l}_n\|_{\mathbb{T}} > 0$. Accordingly, we can ignore the samples with small errors and the backpropagation procedure is significantly accelerated. In this work, we use $\beta = 0.0025$ and $u = 3$.

III. Update CNN for Tracking

A. Online Learning: Iterative SGD With Temporal Sampling

1) Temporal Sampling: Following other CNN-based approaches [12], [13], we used Stochastic Gradient Decent (SGD) for the learning of the parameters Ω. However, the SGD we employ is specifically tailored for visual tracking.

Different from detection and recognition tasks, the training sample pool grows gradually as new frames come in visual tracking. Moreover, it is desired to learn a consistent object model over *all* the previous frames and then use it to distinguish the object from the background in the *current* frame. This implies that we can effectively learn a discriminative model on a long-term positive set and a short-term negative set.

Based on this intuition, we tailor the SGD method by imposing a temporal sampling process. In particular, let $\mathbf{a_n}$ denote the patch feature cropped in frame Γ according to $\mathbf{y_n}$, *i.e.*, $\mathbf{a} = \phi\langle\Gamma, \mathbf{y}_n\rangle$. We generate the positive sample pool as $\mathbb{A}^+_{1:t} = \{\mathbf{a}^+_{1,(1)}, \mathbf{a}^+_{2,(1)}, \ldots, \mathbf{a}^+_{\theta^+-1,(t)}, \mathbf{a}^+_{\theta^+,(t)}\}$ and the negative sample pool as $\mathbb{A}^-_{1:t} = \{\mathbf{a}^-_{1,(1)}, \mathbf{a}^-_{2,(1)}, \ldots, \mathbf{a}^-_{\theta^--1,(t)}, \mathbf{a}^-_{\theta^-,(t)}\}$, where $\theta^{\pm}$ is the number of positive/negative samples added into the pool, for each frame. When generating a mini-batch for SGD, we sample the positive pool with the probability

$$\mathrm{Prob}(\mathbf{a}^+_{n,(t')}) = \frac{1}{N^+}, \tag{11}$$

while sample the negative samples with the probability

$$\mathrm{Prob}(\mathbf{a}^-_{n,(t')}) = \frac{1}{ZN^-}\exp\left[-\sigma(t-t')^2\right], \tag{12}$$

where $\frac{1}{Z}$ is the normalization term and $\sigma = 10, \theta^+ = 1$, $\theta^- = 150$. In this work, we set $N^+ = \min(t\theta^+, 1000)$, $N^- = \min(t\theta^-, 500)$. When the pool size reaches the limit N^+ and N^-, the oldest ones will be removed.

The above temporal selection mechanism can be considered to be similar to the "multiple-lifespan" data sampling [25]. However, [25] builds three different codebooks, each corresponding to a different lifespan, while we learn one discriminative model based on two different sampling distributions.

2) Robust Temporal Sampling With Label Noise: In most tracking-by-detection strategy, the detected object $\mathbf{y}^*_{(t)}$ is treated as a true-positive in the following training stage. However, among all the motion states $\mathbf{y}^*_{(t)}$, $\forall t = 1, 2, \ldots, T$, only the first one $\mathbf{y}^*_{(1)}$ is reliable as it is manually defined. Other motion states are estimated based on the previous observations. Thus, the uncertainty of the prediction $\mathbf{y}_{(t)}$, $\forall t > 1$ is usually unavoidable [19]. Recall that, the structural loss defined in Eq. 4 could change significantly if a minor perturbation is imposed on $\mathbf{y}_{(t)}$, one requires a accurate $\mathbf{y}_{(t)}$ in every frame, which is unfortunately not feasible.

In our previous work [36], we take the uncertainty into account by imposing a robust term on the loss function Eq. 10. The robust term is designed in the principle of Multiple-Instance-Learning [37], [38] and it alleviates over-fittings in some scenarios. However, the positive-sample-bag [36] could also reduce the learning effectiveness as it will confuse the learner when two distinct samples are involved in one bag. Actually, other MIL-based trackers also suffer from this problem [19], [37].

In this work, we propose a much simpler scheme for addressing the issue of prediction uncertainty. Specifically, the prediction uncertainty is treated as a label noise problem [20]–[22]. We assume there exist some frames, on which the detected "objects" are false-positive samples. In other words, the some sample labels in $\mathbb{A}^+_{1:t}$ and $\mathbb{A}^-_{1:t}$ are contaminated (flipped for the binary case). In the context of temporal sampling, the assumption introduces an extra random variable $\boldsymbol{\eta}$ which represent the event that the label is true ($\boldsymbol{\eta} = 1$) or not ($\boldsymbol{\eta} = 0$). The sampling process is now conduct in the joint probability space $\{n = 1, 2, \cdots, N\} \times \{t' = 1, 2, \cdots, t\} \times \{\boldsymbol{\eta} = 1, 0\}$ and the joint probability is

$$\mathrm{Prob}(\mathbf{y}^{\pm}_{n,(t')}, \boldsymbol{\eta} = 1), \tag{13}$$

where $\mathbf{y}^{\pm}_{n,(t')}$ stands for the selection of the n-th positive/negative sample in the t'-th frame. According to the Bayesian chain-rule, we have

$$\begin{aligned}\mathrm{Prob}(\mathbf{y}^{\pm}_{n,(t')}, \boldsymbol{\eta} = 1) &= \mathrm{Prob}(t', n, \boldsymbol{\eta} = 1)\\ &= \mathrm{Prob}(\boldsymbol{\eta} = 1 \mid t', n)\cdot\mathrm{Prob}(t', n)\\ &= \mathrm{Prob}(\boldsymbol{\eta} = 1 \mid t', n)\cdot\mathrm{Prob}\big(\mathbf{y}^{\pm}_{n,(t')}\big),\end{aligned} \tag{14}$$

where $\mathrm{Prob}\left(\mathbf{y}^{\pm}_{n,(t')}\right)$ is given in Eq. 11 and 12 while the conditional probability $\mathrm{Prob}(\boldsymbol{\eta} = 1 \mid t', n)$ reflects the likelihood that the label of sample $\mathbf{y}^{\pm}_{n,(t')}$ is not contaminated.

To estimate $\mathrm{Prob}(\boldsymbol{\eta} = 1 \mid t', n)$ efficiently, we assume that all the windows in the same frame share the same conditional probabilities. Then we can calculate the probability as a prediction quality $\mathrm{Q}_{t'}$ in frame-t', *i.e.*,

$$\begin{aligned}\mathrm{Q}_{t'} &= \mathrm{Prob}(\boldsymbol{\eta} = 1 \mid t', n)\\ &= 1 - \frac{1}{|\mathbb{P}|}\sum_{n\in\mathbb{P}}^{N}\Big[\Delta(\mathbf{y}_{n,(t')}, \mathbf{y}^*_{(t')})\\ &\qquad\cdot\big\|f(\phi\langle\Gamma, \mathbf{y}_{n,(t')}\rangle; \Omega) - \mathbf{l}_n\big\|_{\mathbb{T}}\Big],\end{aligned} \tag{15}$$

where the set $\mathbb{P}$ contains the sample in the frame t' with high scores. Mathematically, it is defined as

$$\mathbb{P} = \{\forall n \mid S_{n,(t')} > \upsilon\cdot S^*_{(t')}\}, \tag{16}$$

where $S_{n,(t')}$ and $S^*_{(t')}$ are the CNN scores (see Eq. 1) of the n-th sample and the sample selected as object in frame t', respectively. The underlying assumption of Eq. 15 is that, a detection heat-map with multiple widely-distributed peaks usually implies low detection quality, as there is only ONE target in the video sequence. This tracking quality is illustrated in Fig. 4. From the figure we can see that when

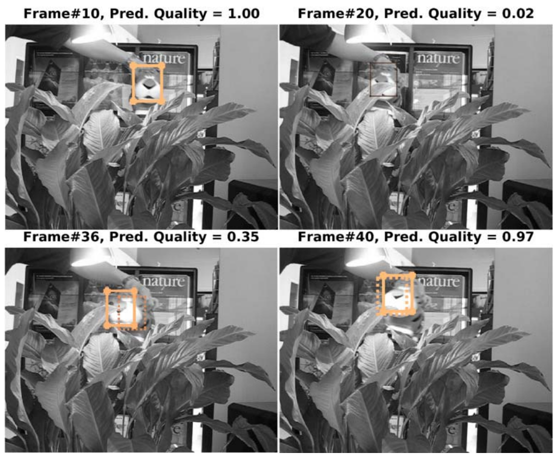

Frame#10, Pred. Quality = 1.00
Frame#20, Pred. Quality = 0.02
Frame#36, Pred. Quality = 0.35
Frame#40, Pred. Quality = 0.97

Fig. 4. A demonstration of the prediction quality on a video sequence (*tiger1*). For each frame, the solid bounding-boxes are the detected object $\mathbf{y}^*_{(t')}$ while the dashed boxes denote the ones with ambiguities, *i.e.*, the members in the set $\mathbb{P}$. For each bounding-box, the importance in the tracking quality evaluation is represented via its width (the wider the more important) and the color (the brighter the more important). The frame index and tracking qualities are shown on the top of the frame images. Note that the quality is estimated without ground-truth information.

occlusion (top-right) or significant appearance change (bottom-left) occurs, the tracking quality drops dramatically and thus the samples in those "contaminated" frames are rarely selected according to Eq. 14. In this work, we set $\upsilon = 0.75$.

3) Iterative Stochastic Gradient Descent (IT-SGD): Recall that we use multiple image cues as the input of the CNN tracker. This leads to a CNN with higher complexity, which implies a low training speed and a high possibility of overfitting. By noticing that each image cue may be weakly independent, we train the network in a iterative manner. In particular, we define the model parameters as $\mathbf{\Omega} = \{\mathbf{w}^1_{cov}, \cdots, \mathbf{w}^K_{cov}, \mathbf{w}^1_{fc}, \cdots, \mathbf{w}^K_{fc}, \mathbf{w}_{fuse}\}$, where $\mathbf{w}^k_{cov}$ denotes the filter parameters in cue-k, $\mathbf{w}^k_{fc}$ corresponds to the fully-connected layers and $\mathbf{w}_{fuse}$ parameterize the fusion layer.

In this work, we conduct the SGD process iteratively over different image cues and the fusion layer. Specifically, after we complete the training on $\mathbf{w}^k_{cov}$ and $\mathbf{w}^k_{fc}$, we evaluate the filter responses from that cue in the last fully-connected layer and then update $\mathbf{w}_{fuse}$ on the dimensions corresponding to cue-k . This can be regarded as a coordinate-descent variation of SGD. In practice, we found out both the robust temporal sampling mechanism and the IT-SGD significantly curb the overfitting problem. The iterative SGD is illustrated in Algorithm 1.

B. Lazy Update and the Overall Work Flow

It is straightforward to update the CNN model using the IT-SGD algorithm at each frame. However, this could be computationally expensive as the complexity of training processes would dominate the complexity of the whole algorithm. On the other hand, in case the appearance of the object is not always changing, a well-learned appearance model can remain discriminant for a long time. Furthermore, when the feature representations are updated for adapting the appearance changes, different image cues contribute to the final classification task in a more stable way.

Algorithm 1 Iterative SGD With Robust Temporal Sampling

```
1:  **Inputs:** Frame image $\Gamma_{(t)}$; Two sample pools $\mathbb{A}^+_{1:t}$, $\mathbb{A}^-_{1:t}$;
2:  Old CNN model ($K$ cues) $f_0(\phi\langle\Gamma_{(t)},\cdot\rangle;\Omega)$.
3:  Estimated/given $\mathbf{y}^*_{(t)}$;
4:  Learning rates $r$; $\hat{r}$; minimal loss $\varepsilon$; training step budget $M$.
5:  **procedure** IT-SGD($\mathbb{A}^+_{1:t}$, $\mathbb{A}^-_{1:t}$, $f$, $\mathbf{y}^*$, $\hat{r}$, $r$, $M$)
6:      Selected sample states $\{\mathbf{y}_{1,(t)}, \mathbf{y}_{2,(t)}, \ldots, \mathbf{y}_{N,(t)}\}$.
7:      Generate associated labels $\mathbf{l}_{1,(t)}, \cdots, \mathbf{l}_{N,(t)}$ according to $\mathbf{y}^*_{(t)}$.
8:      Estimate the prediction quality $\mathrm{Q}_t$.
9:      Save the current samples and labels into $\mathbb{A}^+_{1:t}$ and $\mathbb{A}^-_{1:t}$.
10:     Sample training instances according to Prob($\mathbf{y}^\pm_{n,(t)}, \boldsymbol{\eta} = 1$).
11:     **for** $m \leftarrow 0,\ M-1$ **do**
12:         $\mathcal{L}_m = \frac{1}{N}\sum_{n=1}^{N}\left[\Delta(\mathbf{y}_n,\mathbf{y}^*)\cdot\left\|f_m(\phi\langle\Gamma_{(t)},\mathbf{y}_n\rangle;\Omega) - \mathbf{l}_n\right\|_{\mathbb{T}}\right]$;
13:         **If** $\mathcal{L}_m \le \varepsilon$, **break**;
14:         $k = \mathrm{mod}(m, K) + 1$;
15:         Update $\mathbf{w}^k_{cov}$ and $\mathbf{w}^k_{fc}$ using SGD with learning rate $r$.
16:         Update $\mathbf{w}_{fuse}$ partially for cue-$k$, with learning rate $\hat{r}$.
17:         Save $f_{m+1} = f_m$;
18:     **end for**
19: **end procedure**
20: **Outputs:** New CNN model $f^* = f_{m^*}, m^* = \arg\min_m \mathcal{L}_m$.
```

Motivated by the above two intuitions, we propose to update the CNN model in a lazy manner (see Algorithm 1). First, when tracking the object, we only update the CNN model when the training loss $\mathcal{L}_1$ is above 2ε. Once the training start, the training goal is to reduce $\mathcal{L}$ below ε. As a result, usually $\mathcal{L}_1 < 2\varepsilon$ holds in a number of the following frames, and thus no training is required for those frames. This way, we accelerate the tracking algorithm significantly (Fig. 5). Second, we update the fusion layer in a lazy way, or more specifically, in a coordinate-descent manner with a small learning rate (see Algorithm 1). The whole learning process over all the image cues is thus stabilized well. In this work, we set that ε = 5e-3, r = 5e-2 and $\hat{r}$ = 5e-3.

C. Modifications to the Previous Versions of the DeepTrack Algorithm

In our previous work [36], [39], we proposed to use a "CNN pool" (a set of CNNs) [39], or a single CNN [36] for visual tracking. A number of modifications have been made on the previous versions of DeepTrack to achieve a more accurate and robust visual tracker. We summarize the differences among the three variations of DeepTrack in Table I. According to the table, the latest version of DeepTrack is more sophisticatedly designed and can lead to better performances, as shown in Sec. IV.

IV. Experiments

A. Benchmarks and Experiment Setting

We evaluate our method on two recently proposed visual tracking benchmarks, *i.e.*, the CVPR2013 Visual Tracker Benchmark [40] and the VOT2013 Challenge Benchmark [41]. These two benchmarks involve more than 60 sequences and cover almost all the challenging scenarios such as scale

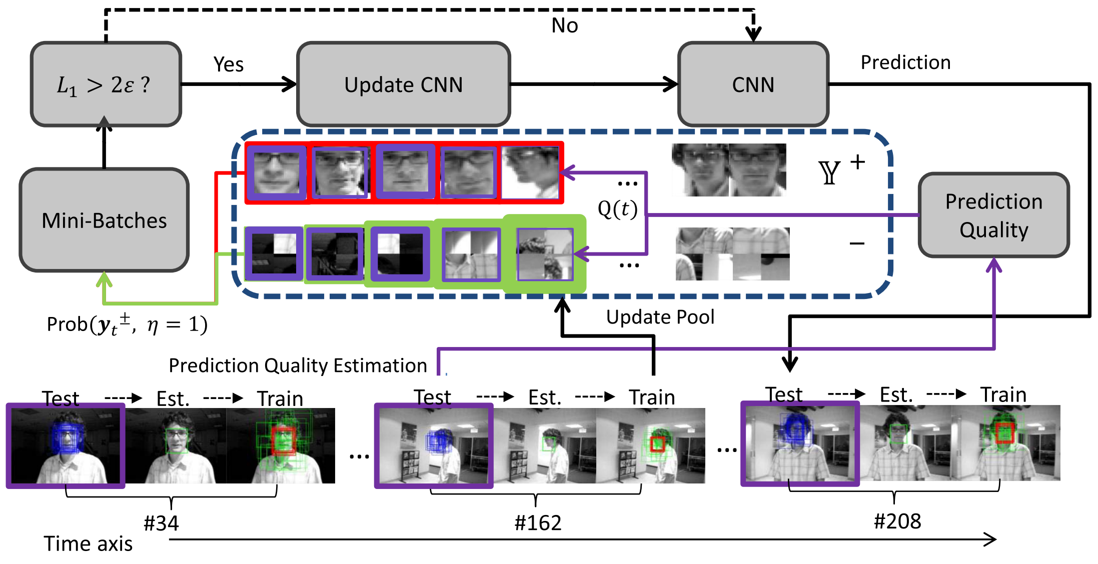


Fig. 5. Work flow of proposed algorithm. The bottom row shows the three-stages operations on a frame: test, estimation and training. In the training frames, the green bounding boxes are the negative samples while the red ones denote the positive samples. The dashed block covers the positive sample pool $\mathbb{A}^+$ (red) and negative sample pool $\mathbb{A}^-$ (green). In each pool, the edges of the sample patches indicate their sampling importances. The green ones (negative) and red ones (positive) represent the prior probabilities of sample selection while the purple ones stands for the conditional probabilities (Q(t)). The thicker the edge, the higher the probability.

TABLE I

THE IMPLEMENTATION DETAILS AND DIFFERENCES OF THREE VERSIONS OF DEEPTRACK. FOR THE ITEM "CNN STRUCUTURE", THE SYMBOL "C", "P" AND "FC" DENOTE THE CONVOLUTIONAL LAYERS, POOLING LAYERS AND FULLY-CONNECTED LAYERS RESPECTIVELY. "9C" MEANS THAT THERE ARE 9 FILTERS IN THIS CONVOLUTIONAL LAYER. "FC(18 × 2)" INDICATES THAT THE MAPPING MATRIX W IN THIS FULLY-CONNECTED LAYER IS $W \in \mathcal{R}^{18\times 2}$

	DeepTrack (this work)	DeepTrack_ACCV[36]	DeepTrack_BMVC[39]
Main Structure	Single CNN	Single CNN	Multiple (50) CNNs
CNN Strucuture	12C→P→ 18C→P→FC(72 × 8)→FC(8 × 2)	9C→P→ 18C→P→FC(18 × 2)	9C→P→ 18C→P→FC(18 × 2)
Pooling method	Max	Average	Average
Activation method	ReLU	Sigmoid	Sigmoid
Image cue number	3	4	4
Data augmentation?	Yes, flipped positive samples	No	No
Cue normalization?	Yes	No	No
Multiple-lifespan sampling?	Yes	Yes	No
Sample with label uncertainty?	Yes	No	No
Truncated structural loss	Improved	Original	Original

changes, illumination changes, occlusions, cluttered backgrounds and motion blur. Furthermore, these two benchmarks evaluate tracking algorithms with different measures and criteria, which can be used to analyze the tracker from different perspectives. Fig. 6 illustrates some sequence instances of the CVPR2013 benchmark (top row) and the VOT2013 benchmark (bottom row), with the objects annotated in the first frames.

In the experiments on two selected benchmarks, we use the same parameter values for DeepTrack. Most parameters of the CNN tracker are given in Sec. II and Sec. III. In addition, there are some motion parameters for sampling the image patches. In this work, we only consider the displacement Δ_x, Δ_y and the relative scale s of the object, where $s = h/32$, h is estimated object height. In a new frame, we sample 1500 random patches in a Gaussian Distribution which centers on the previous predicted state. The standard deviation for the three dimensions are $\min(10, 0.5 \cdot h)$, $\min(10, 0.5 \cdot h)$ and $0.01 \cdot h$, respectively. Note that, all parameters are fixed for all videos in both two benchmarks; no parameter tuning is performed for any specific video sequence. We run our algorithm in Matlab with an unoptimized code mixed with CUDA-PTX kernels for the CNN implementation. The hardware environment includes one quad-core CPU and a NVIDIA GTX980 GPU.

B. Comparison Results on the CVPR2013 Benchmark

The CVPR2013 Visual Tracker Benchmark [40] contains 50 fully annotated sequences. These sequences include many popular sequences used in the online tracking literature over the past several years. For better evaluation and analysis of the strength and weakness of tracking approaches, these sequences are annotated with the 11 attributes including illumination variation, scale variation, occlusion, deformation, motion blur,

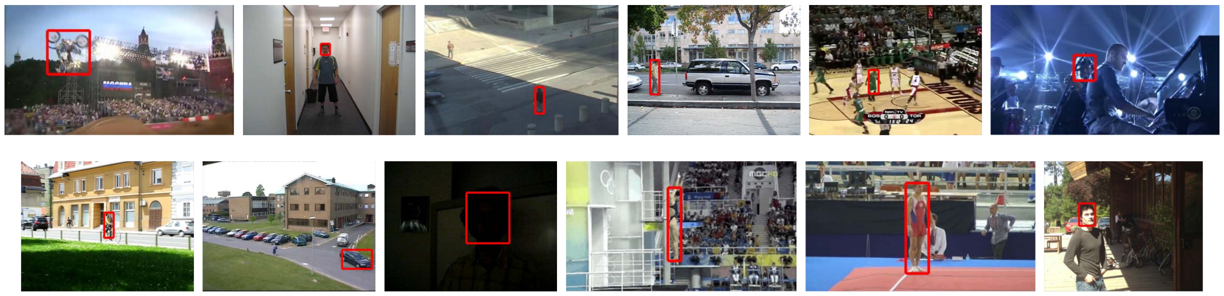


Fig. 6. The first frames of the video sequences from the CVPR2013 (first row) and the VOT2013 (second row) benchmarks. From top left to bottom right: MotorRolling, Boy, Crossing, David3, Basketball, Shaking, Bicycle, Car, David, Diving, Gymnastics, Sunshade. The red blocks are the object locations given in the first frame.

fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution. The benchmark contains the results of 29 tracking algorithms published before the year 2013. Here, we compare our method with other 11 tracking methods. Among the competitors, CNT [42], TPGR [43] and KCF [44] are the most recently state-of-the-art visual trackers; TLD [45], VTD [46], CXT [47], ASLA [48], Struck [4], SCM [49] are the top-6 methods as reported in the benchmark; IVT [31] and MIL [19] are classical tracking methods which are used as comparison baselines. In addition, to evaluate the performance improvement to the previous versions of DeepTrack, we also rerun the DeepTrack method proposed in [39] (DeepTrack_BMVC) and [36] (DeepTrack_ACCV) report the obtained results here.

The tracking results are evaluated via the following two measurements: 1) Tracking Precision (TP), the percentage of the frames whose estimated location is within the given distance-threshold (τ_d) to the ground truth, and 2) Tracking Success Rate (TSR), the percentage of the frames in which the overlapping score defined in Eq. 4 between the estimated location and the ground truth is larger than a given overlapping-threshold (τ_o). Following the setting in the recently published work [43], [44], we conduct the experiment using the OPE (one-pass evaluation) evaluation strategy for a better comparison to the latest methods.

Firstly, we evaluate all algorithms using fixed thresholds, *i.e.*, $\tau_d = 20$, $\tau_o = 0.6$, which is a common selection in tracking evaluations [40]. Results of the top-12 trackers (all the involved ones except IVT and MIL trackers) on all the video sequences are given in Table II. According to the table, our method achieves better average performance compared with other trackers. The performance gap between our method and the reported best result in the literature are 6% for the TP measure: our method achieves 83% accuracy while the best state-of-the-art is 77% (TGPR method). For the TSR measure, our method is 8% better than the existing methods: our method gives 63% accuracy while the best state-of-the-art is 58% (CNT method). Furthermore, our CNN tracker have ranked as the best method for 33 times. These numbers for CNT, TGPR, KCF, SCM and Struck are 29, 21, 28, 19 and 21 respectively. Another observation from the Table II is that, DeepTrack rarely performs inaccurately; there are only 36 occasions when the proposed tracker performs significantly poorer than the best method (no less then 80% of the highest score for one sequence).

The superiority of our method becomes more clear when the tracking result are evaluated using different measurement criteria (different τ_d, τ_o). In specific, for TP, we evaluate the trackers with the thresholds $\tau_d = 1, 2, \cdots, 50$ while for TSR, we use the thresholds $\tau_o = 0$ to 1 at the step of 0.05. Accordingly we generate the precision curves and the success-rate curves for each tracking method, which is shown in Fig. 7.

From the score plots we can see that, overall the CNN tracker ranks the first (red curves) for both TP and TSR evaluations. The proposed DeepTrack method outperform all the other trackers when $\tau_o < 0.68$ and $\tau_d > 10$. When the evaluation threshold is reasonably loose, (*i.e.*, $\tau_o < 0.45$ and $\tau_d > 20$), our algorithm is very robust with both the accuracies higher than 80%. Having mentioned that when the overlap thresholds are tight (*e.g.* $\tau_o > 0.75$ or $\tau_d < 5$), our tracker has similar response to rest of the trackers we tested.

As to the different variations of DeepTrack, one can see that the previous versions, *i.e.*, DeepTrack_BMVC and DeepTrack_ACCV, can achieve relatively good accuracies which are comparable to the top ranked trackers (*e.g.*, Struck and SCM) reported with the benchmark. On the other hand, the current version of DeepTrack still enjoys a large performance improvement over its prototypes.

Fig. 8 shows the performance plots for 11 types of difficulties in visual tracking, *i.e.*, *fast-motion, background-clutter, motion-blur, deformation, illumination-variation, in-plane-rotation, low-resolution, occlusion, out-of-plane-rotation, out-of-view and scale-variations*. We can see that the proposed DeepTrack outperforms other competitors for all the difficulties except the "out-of-view" category.

C. Comparison Results on the VOT2013 Benchmark

The VOT2013 Challenge Benchmark [41] provides an evaluation kit and the dataset with 16 fully annotated sequences for evaluating tracking algorithms in realistic scenes subject to various common conditions. The tracking performance in the VOT2013 Challenge Benchmark is primarily evaluated with two evaluation criteria: accuracy and robustness. The accuracy measure is the average of the overlap ratios over the valid frames of each sequence while the tracking

TABLE II

THE TRACKING SCORES OF DEEPTRACK AND OTHER VISUAL TRACKERS ON THE CVPR2013 BENCHMARK. THE REPORTED RESULTS ARE SHOWN IN THE ORDER OF "TP/TSR". THE TOP SCORES ARE SHOWN IN RED FOR EACH ROW. A SCORE IS SHOWN IN BLUE IF IT IS HIGHER THAN 80% OF THE HIGHEST VALUE IN THAT ROW. "NO. BEST" ROW SHOWS THE NUMBER OF BEST SCORES FOR EACH TRACKING ALGORITHM WHILE "NO. BAD" ROW SHOWS THE NUMBER OF LOW SCORES, *i.e.*, THE SCORES LOWER THAN 80% OF THE MAXIMUM ONE IN THE CORRESPONDING ROW. ASIDE THE CURRENT VERSION OF DEEPTRACK, WE ALSO REPORT THE RESULTS OF ITS OLD VERSIONS PROPOSED IN [39] (BMVC) AND [36] (ACCV)

	Struck	VTD	CXT	SCM	TLD	ASLA	KCF	TGPR	CNT	BMVC	ACCV	DeepTrack
tiger1	0.17/0.13	0.12/0.09	0.37/0.17	0.13/0.11	0.46/0.36	0.23/0.15	0.97/0.94	0.28/0.22	0.15/0.12	0.21/0.06	0.10/0.05	0.56/0.36
carDark	1.00/1.00	0.74/0.66	0.73/0.67	1.00/0.98	0.64/0.50	1.00/0.99	1.00/0.44	1.00/0.95	1.00/0.99	1.00/0.78	1.00/0.94	1.00/0.97
girl	1.00/0.90	0.95/0.41	0.77/0.61	1.00/0.74	0.92/0.61	1.00/0.78	0.86/0.47	0.92/0.69	1.00/0.36	0.96/0.20	0.92/0.64	0.98/0.83
david	0.33/0.19	0.94/0.38	1.00/0.48	1.00/0.84	1.00/0.83	1.00/0.94	1.00/0.26	0.98/0.26	0.47/0.32	0.96/0.89	0.79/0.74	1.00/0.76
singer1	0.64/0.20	1.00/0.36	0.97/0.27	1.00/1.00	1.00/0.93	1.00/0.98	0.81/0.20	0.68/0.19	1.00/1.00	0.75/0.61	0.74/0.55	1.00/1.00
skating1	0.47/0.20	0.90/0.43	0.23/0.06	0.77/0.21	0.32/0.21	0.77/0.45	1.00/0.23	0.81/0.25	1.00/0.43	0.76/0.32	0.70/0.17	1.00/0.45
deer	1.00/0.94	0.04/0.03	1.00/0.87	0.03/0.03	0.73/0.73	0.03/0.03	0.82/0.76	0.86/0.79	1.00/0.89	0.83/0.82	0.73/0.71	1.00/0.99
singer2	0.04/0.03	0.45/0.43	0.06/0.04	0.11/0.13	0.07/0.05	0.04/0.03	0.95/0.89	0.97/0.91	0.95/0.97	0.91/0.83	0.92/0.86	0.57/0.34
car4	0.99/0.26	0.36/0.32	0.38/0.27	0.97/0.93	0.87/0.63	1.00/0.95	0.95/0.24	1.00/0.28	1.00/1.00	1.00/0.84	1.00/0.99	1.00/1.00
tiger2	0.63/0.42	0.16/0.08	0.34/0.16	0.11/0.05	0.39/0.04	0.14/0.11	0.36/0.28	0.72/0.47	0.11/0.09	0.41/0.17	0.24/0.17	0.49/0.32
dudek	0.90/0.81	0.88/0.96	0.82/0.87	0.88/0.86	0.60/0.63	0.75/0.74	0.88/0.82	0.75/0.71	0.87/0.85	0.50/0.64	0.25/0.25	0.73/0.81
sylvester	0.99/0.85	0.82/0.74	0.85/0.56	0.95/0.77	0.95/0.80	0.82/0.65	0.84/0.73	0.96/0.93	0.89/0.55	0.97/0.71	0.97/0.31	1.00/0.92
jumping	1.00/0.50	0.21/0.08	1.00/0.25	0.15/0.11	1.00/0.70	0.45/0.15	0.34/0.26	0.95/0.50	1.00/0.31	0.99/0.81	1.00/0.87	1.00/0.93
david2	1.00/1.00	1.00/0.88	1.00/1.00	1.00/0.80	1.00/0.70	1.00/0.95	1.00/1.00	1.00/0.97	1.00/1.00	1.00/0.79	1.00/0.98	1.00/0.87
shaking	0.19/0.04	0.93/0.83	0.13/0.04	0.81/0.69	0.41/0.31	0.48/0.17	0.02/0.01	0.97/0.70	0.01/0.01	0.32/0.23	0.58/0.45	0.95/0.68
trellis	0.88/0.72	0.50/0.44	0.97/0.69	0.87/0.84	0.53/0.45	0.86/0.85	1.00/0.74	0.98/0.68	1.00/0.95	1.00/0.80	0.99/0.88	1.00/0.96
woman	1.00/0.89	0.20/0.16	0.37/0.15	0.94/0.69	0.19/0.15	0.20/0.17	0.94/0.90	0.97/0.87	0.94/0.87	0.20/0.07	0.16/0.05	0.98/0.24
fish	1.00/1.00	0.65/0.57	1.00/1.00	0.86/0.85	1.00/0.96	1.00/1.00	1.00/1.00	0.97/0.97	1.00/1.00	1.00/0.98	1.00/0.99	1.00/1.00
matrix	0.12/0.12	0.22/0.03	0.06/0.01	0.35/0.24	0.16/0.03	0.05/0.01	0.17/0.11	0.39/0.26	0.16/0.14	0.26/0.15	0.37/0.15	0.72/0.43
ironman	0.11/0.02	0.17/0.12	0.04/0.03	0.16/0.09	0.12/0.04	0.13/0.08	0.22/0.10	0.22/0.13	0.19/0.08	0.13/0.09	0.15/0.11	0.08/0.05
mhyang	1.00/0.97	1.00/0.77	1.00/1.00	1.00/0.96	0.98/0.52	1.00/1.00	1.00/0.93	0.95/0.88	1.00/1.00	0.94/0.66	1.00/0.98	1.00/0.96
liquor	0.39/0.40	0.52/0.52	0.21/0.21	0.28/0.29	0.59/0.54	0.23/0.23	0.98/0.97	0.27/0.27	0.48/0.48	0.26/0.20	0.47/0.42	0.91/0.89
motorRolling	0.09/0.09	0.05/0.05	0.04/0.02	0.04/0.05	0.12/0.10	0.06/0.07	0.05/0.05	0.09/0.10	0.04/0.05	0.04/0.06	0.07/0.10	0.80/0.43
coke	0.95/0.87	0.15/0.11	0.65/0.15	0.43/0.24	0.68/0.09	0.16/0.10	0.84/0.41	0.95/0.63	0.52/0.18	0.22/0.10	0.21/0.06	0.91/0.18
soccer	0.25/0.15	0.45/0.18	0.23/0.12	0.27/0.16	0.11/0.11	0.12/0.11	0.79/0.35	0.16/0.13	0.12/0.09	0.17/0.15	0.17/0.14	0.30/0.16
boy	1.00/0.93	0.97/0.61	0.94/0.42	0.44/0.44	1.00/0.74	0.44/0.44	1.00/0.96	0.99/0.91	1.00/0.99	1.00/0.86	1.00/0.95	1.00/0.93
basketball	0.12/0.09	1.00/0.85	0.04/0.02	0.66/0.53	0.03/0.02	0.60/0.26	0.92/0.71	0.99/0.69	0.07/0.05	0.40/0.03	0.47/0.12	0.82/0.39
lemming	0.63/0.49	0.51/0.42	0.73/0.38	0.17/0.16	0.86/0.43	0.17/0.17	0.49/0.30	0.35/0.26	0.38/0.37	0.60/0.37	0.80/0.71	0.28/0.26
bolt	0.02/0.01	0.31/0.14	0.03/0.01	0.03/0.01	0.31/0.08	0.02/0.01	0.99/0.75	0.02/0.01	0.98/0.45	0.03/0.01	0.03/0.01	0.99/0.78
crossing	1.00/0.72	0.44/0.36	0.62/0.32	1.00/0.99	0.62/0.41	1.00/0.99	1.00/0.78	1.00/0.81	1.00/0.96	0.78/0.19	1.00/0.80	0.94/0.56
couple	0.74/0.51	0.11/0.06	0.64/0.52	0.11/0.11	1.00/0.98	0.09/0.09	0.26/0.24	0.60/0.35	0.23/0.21	0.62/0.22	0.51/0.16	0.99/0.63
david3	0.34/0.34	0.56/0.44	0.15/0.10	0.50/0.47	0.11/0.10	0.55/0.49	1.00/0.96	1.00/0.69	0.65/0.39	0.29/0.09	0.42/0.12	1.00/0.93
carScale	0.65/0.37	0.55/0.42	0.74/0.74	0.65/0.64	0.85/0.29	0.74/0.65	0.81/0.35	0.79/0.37	0.72/0.63	0.75/0.27	0.63/0.34	0.67/0.56
doll	0.92/0.34	0.97/0.73	0.99/0.87	0.98/0.97	0.98/0.39	0.92/0.91	0.97/0.33	0.94/0.40	0.95/0.92	0.98/0.91	0.99/0.94	0.96/0.86
skiing	0.04/0.04	0.14/0.01	0.15/0.06	0.14/0.06	0.12/0.05	0.14/0.11	0.07/0.05	0.12/0.10	0.07/0.06	0.07/0.06	0.05/0.05	0.09/0.06
football	0.75/0.57	0.80/0.65	0.80/0.57	0.77/0.42	0.80/0.28	0.73/0.62	0.80/0.57	1.00/0.75	0.80/0.57	0.85/0.32	0.86/0.45	0.79/0.52
football1	1.00/0.72	0.99/0.51	1.00/0.96	0.57/0.34	0.55/0.34	0.80/0.39	0.96/0.80	0.99/0.41	0.99/0.34	0.94/0.34	0.86/0.36	1.00/0.38
freeman1	0.80/0.16	0.95/0.13	0.73/0.18	0.98/0.54	0.54/0.18	0.39/0.20	0.39/0.13	0.93/0.21	0.96/0.26	0.98/0.17	0.99/0.30	1.00/0.35
freeman3	0.79/0.12	0.72/0.22	1.00/0.89	1.00/0.88	0.77/0.42	1.00/0.90	0.91/0.21	0.77/0.15	1.00/0.68	0.71/0.21	0.25/0.11	0.97/0.67
freeman4	0.37/0.15	0.37/0.08	0.43/0.17	0.51/0.18	0.41/0.24	0.22/0.16	0.53/0.12	0.58/0.21	0.39/0.15	0.66/0.16	0.23/0.11	0.71/0.22
subway	0.98/0.63	0.23/0.18	0.26/0.20	1.00/0.90	0.25/0.22	0.23/0.21	1.00/0.94	1.00/0.99	1.00/0.62	0.93/0.22	0.80/0.69	1.00/0.79
suv	0.57/0.57	0.52/0.47	0.91/0.90	0.98/0.80	0.91/0.70	0.57/0.55	0.98/0.98	0.66/0.66	0.98/0.97	0.53/0.52	0.53/0.53	0.52/0.52
walking	1.00/0.42	1.00/0.55	0.24/0.22	1.00/0.86	0.96/0.30	1.00/0.99	1.00/0.34	1.00/0.41	1.00/1.00	1.00/0.04	1.00/0.36	1.00/0.94
walking2	0.98/0.32	0.41/0.39	0.41/0.39	1.00/0.99	0.43/0.29	0.40/0.40	0.44/0.30	0.99/0.31	1.00/1.00	0.96/0.81	0.57/0.53	0.61/0.38
mountainBike	0.92/0.67	1.00/0.81	0.28/0.28	0.97/0.72	0.26/0.21	0.90/0.82	1.00/0.88	1.00/0.87	1.00/0.86	0.92/0.27	0.92/0.33	1.00/0.91
faceocc1	0.58/0.95	0.53/0.72	0.34/0.57	0.93/1.00	0.20/0.65	0.18/0.25	0.73/0.99	0.66/0.80	0.65/0.86	0.70/0.75	0.68/0.75	0.33/0.42
jogging-1	0.24/0.22	0.23/0.18	0.96/0.95	0.23/0.21	0.97/0.95	0.23/0.22	0.23/0.22	0.99/0.96	0.97/0.49	0.69/0.13	0.83/0.24	0.97/0.94
jogging-2	0.25/0.22	0.19/0.16	0.16/0.15	1.00/0.98	0.86/0.83	0.18/0.17	0.16/0.15	1.00/0.95	1.00/1.00	0.18/0.15	0.17/0.16	0.99/0.30
dog1	1.00/0.51	0.83/0.61	1.00/0.95	0.98/0.76	1.00/0.61	1.00/0.87	1.00/0.51	1.00/0.52	0.94/0.82	0.96/0.59	0.99/0.63	1.00/0.95
fleetface	0.64/0.51	0.66/0.68	0.57/0.60	0.53/0.58	0.51/0.41	0.30/0.32	0.46/0.47	0.45/0.47	0.59/0.62	0.21/0.22	0.30/0.25	0.51/0.60
faceocc2	1.00/0.97	0.98/0.84	1.00/0.90	0.86/0.74	0.86/0.51	0.79/0.61	0.97/0.79	0.47/0.45	0.65/0.50	0.97/0.78	0.98/0.83	1.00/0.71
Overall	**0.66/0.48**	**0.58/0.41**	**0.58/0.43**	**0.65/0.55**	**0.61/0.42**	**0.53/0.46**	**0.74/0.53**	**0.77/0.54**	**0.72/0.58**	**0.66/0.41**	**0.64/0.46**	**0.83/0.63**
No. Best	**21**	**10**	**15**	**18**	**12**	**17**	**27**	**20**	**29**	**4**	**7**	**33**
No. Bad	**62**	**71**	**66**	**51**	**72**	**64**	**48**	**45**	**44**	**67**	**65**	**36**

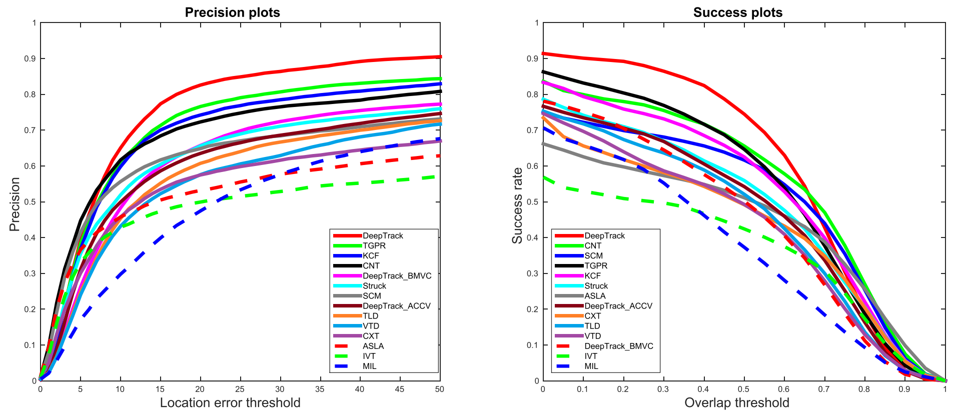


Fig. 7. The Precision Plot (left) and the Success Plot (right) of the tracking results on the CVPR2013 benchmark. Note that the color of one curve is determined by the rank of the corresponding trackers, not their names.

robustness is the average number of failures over 15 runs. A tracking failure happens once the overlap ratio measure drops to zero and an re-initialization of the tracker in the failure frame is conducted so it can continue. According to the evaluation protocol, three types of experiments are conducted. In Experiment-1, the tracker is run on each sequence in

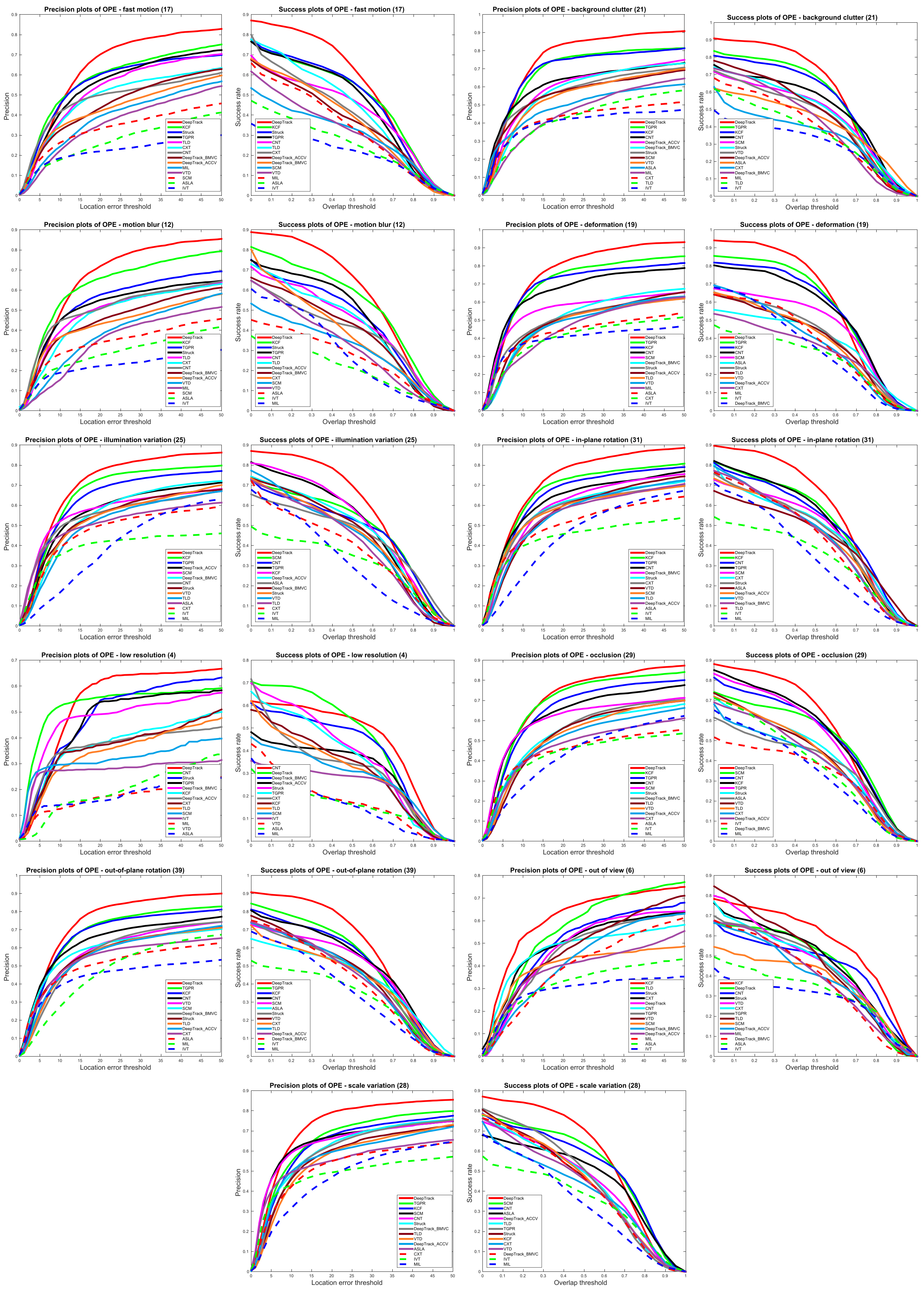


Fig. 8. The Precision Plot (left) and the Success Plot (right) of the tracking results on the CVPR2013 benchmark, for 11 types of tracking difficulties.

the dataset 15 times by initializing it on the ground truth bounding box. The setting of Experiment-2 is the same to Experiment-1, except that the initial bounding box is randomly perturbed in the order of ten percent of the object size. In Experiment-3, the colorful frames are converted into grayscale images.

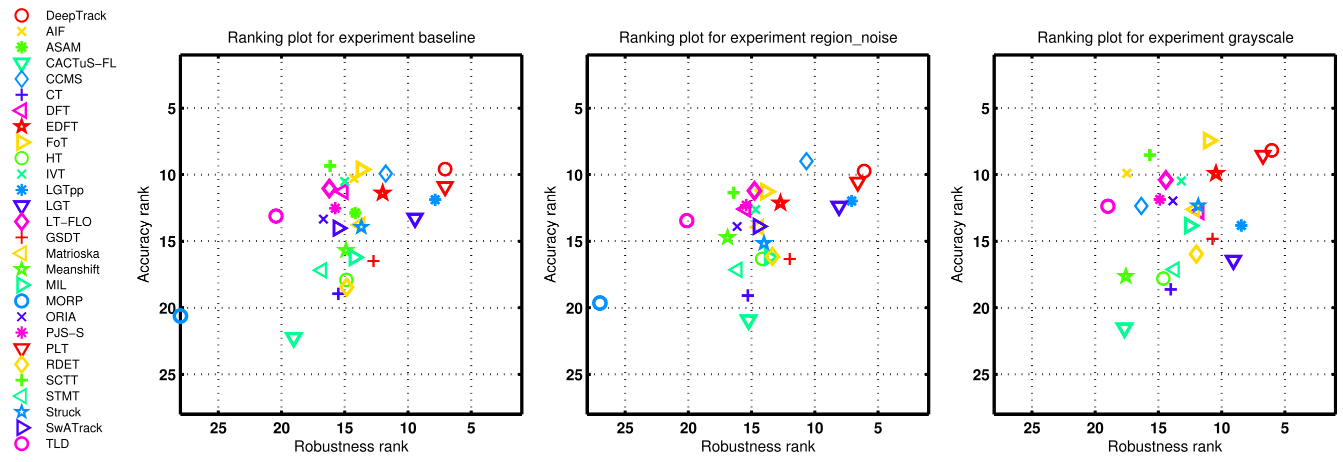


Fig. 9. The Precision Plot (left) and the Success Plot (right). The color of one curve is determined by the rank of the corresponding trackers, not their names.

TABLE III
THE PERFORMANCE COMPARISON BETWEEN CNN TRACKER AND OTHER 27 TRACKERS ON THE VOT2013 BENCHMARK. FOR EACH COLUMN, THE BEST SCORE IS SHOWN IN BOLD AND RED WHILE THE SECOND BEST SCORE IS SHOWN IN BLUE

	Experiment-1		Experiment-2		Experiment-3		Averaged	
	Accu.	Rob.	Accu.	Rob.	Accu.	Rob.	Accu.	Rob.
DeepTrack	9.60	**7.06**	10.14	**6.09**	8.17	**6.04**	**9.30**	**6.40**
AIF	10.29	14.27	11.39	14.43	9.90	17.50	10.52	15.40
ASAM	12.86	14.17	NaN	NaN	NaN	NaN	NaN	NaN
CACTuS-FL	22.27	19.03	20.92	15.24	21.54	17.69	21.58	17.32
CCMS	9.87	11.76	**8.97**	10.66	12.36	16.35	10.40	12.92
CT	18.92	15.51	19.10	15.30	18.62	14.03	18.88	14.95
DFT	11.23	15.12	12.64	15.47	12.78	11.79	12.21	14.13
EDFT	11.45	12.00	12.18	12.72	9.91	10.44	11.18	11.72
FoT	9.63	13.69	11.23	13.87	**7.44**	11.04	9.44	12.87
HT	17.90	14.88	16.29	14.17	17.81	14.61	17.33	14.55
IVT	10.51	15.00	12.66	14.68	10.48	13.19	11.21	14.29
LGTpp	11.89	7.84	11.98	7.08	13.82	8.44	12.56	7.79
LGT	13.27	9.44	12.37	8.08	16.43	9.07	14.02	8.86
LT-FLO	11.09	16.22	11.25	14.77	10.40	14.41	10.91	15.14
GSDT	16.51	12.73	16.36	11.98	14.82	10.73	15.90	11.81
Matrioska	13.77	13.78	13.94	14.43	12.60	12.13	13.44	13.45
Meanshift	15.69	14.91	14.82	16.90	17.64	17.57	16.05	16.46
MIL	16.38	14.25	16.28	13.58	13.84	12.54	15.50	13.46
MORP	20.64	28.00	19.65	27.00	NaN	NaN	NaN	NaN
ORIA	13.13	16.69	13.86	16.15	11.97	13.85	12.99	15.56
PJS-S	12.50	15.75	12.31	15.43	11.87	14.89	12.22	15.36
PLT	10.88	**7.06**	10.58	6.60	8.54	6.73	10.00	6.79
RDET	18.42	14.84	16.14	13.35	15.97	12.00	16.84	13.40
SCTT	**9.36**	16.16	11.37	16.43	8.53	15.68	9.75	16.09
STMT	17.16	16.81	17.17	16.12	17.12	13.73	17.15	15.55
Struck	13.92	13.69	15.21	14.02	12.33	11.85	13.82	13.19
SwATrack	13.98	15.53	13.93	14.48	NaN	NaN	NaN	NaN
TLD	13.12	20.44	13.37	20.12	12.37	19.00	12.95	19.85

Firstly, we follow the evaluation protocol to test our method, compared with other 27 tracking algorithms provided in the benchmark website. The main comparison results can be found in Table III and Fig. 9. We can see that, in average, the proposed method ranks the first for both accuracy and robustness comparison. In specific, DeepTrack achieves the best robustness scores for all the scenarios while ranks the second in accuracy for all the experimental settings. In the Fig. 9, one can observe that the red circles (which stands for DeepTrack) always locate in the top-right corner of the plot. This observation is consistent to the scores reported in Table III. From the result we can see that our DeepTrack achieves close while consistently better performances than the PLT method [41]. Other tracking methods that can achieve similar performances on this benchmarks are FoT [50], EDFT [51] and LGT++ [52].

Note that the scores listed in Table III and the plots in Fig. 9 are rank-based, which is different from the measuring criterion used in the CVPR2013 benchmark. It is well-known that the evaluation method for visual tracker is not unique and could be sophisticated for a specific objective [53]. Usually different tracker measures offer different points of view for accessing the tracking method. The best performance on the VOT2013 benchmark justifies the superiority of DeepTrack, from another perspective.

In [43], the authors run their TGPR tracker on the VOT2013 benchmark, without comparing with other trackers. We here compare our DeepTrack with the TGPR algorithm, which is recently proposed and achieves state-of-the-art performance in the CVPR2013 benchmark. Following the settings in [43], we perform the proposed tracker in Experiment-1 and Experiment-2. The performance comparison is shown in Table IV.

We can see that the proposed DeepTrack outperforms the TPGR tracker in the robustness evaluation, with a clear performance gap. For Experiment-1, one needs to reinitialize the TPGR tracker for 0.71 times per sequence while that number for our method is 0.22. Similarly, with the bounding box perturbation (Experiment-2), TPGR needs 0.73 times re-initialization while DeepTrack still requires 0.22 times. Note that in Table IV the accuracies from different trackers are not directly comparable, as they are calculated based on different re-initialization conditions. However, by observing the overall scores, we can still draw the conclusion that the DeepTrack is more robust than TPGR as it achieves similar accuracies to TPGR (0.62 *v.s.* 0.64 for Experiment-1 and 0.59 *v.s.* 0.58 for Experiment-2) while only requires around one third of re-initializations.

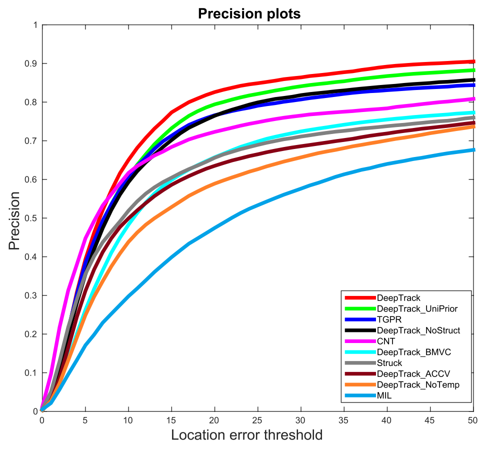
D. The Verification for the Proposed Modifications

Here we verify the three proposed modifications to the CNN model. We rerun the experiment on the CVPR2013 benchmark using the DeepTrack with each modification inactivated. In specific, the temporal sampling mechanism, the

TABLE IV

THE PERFORMANCE COMPARISON BETWEEN DEEPTRACK TRACKER AND THE TPGR TRACKER ON THE VOT2013 BENCHMARK. THE BETTER ROBUSTNESS SCORE IS SHOWN IN BOLD. NOTE THAT FOR ACCURACY (ACCU.), THE COMPARISON IS NOT FAIR IF THE ROBUSTNESS SCORE IS DIFFERENT AND THUS NO BOLD ACCURACY SCORE IS SHOWN

	bicycle	*bolt*	*car*	*cup*	*david*	*diving*	*face*	*gym*	*hand*	*iceskater*	*juice*	*jump*	*singer*	*sunshade*	*torus*	*woman*	overall
Exp1-TPGR-Rob.	**0**	1.27	**0.40**	**0**	0.27	2.87	**0**	2.87	1.67	**0**	**0**	**0**	0.60	0.20	0.13	1.00	0.71
Exp1-DeepTrack-Rob.	0.47	**0.07**	0.47	0	**0.20**	**0.80**	**0**	**0.73**	**0.20**	**0**	**0**	**0**	**0**	**0**	**0.07**	**0.47**	**0.22**
Exp1-TPGR-Accu.	0.60	0.57	0.45	0.83	0.58	0.33	0.85	0.57	0.56	0.60	0.76	0.59	0.65	0.73	0.78	0.74	0.64
Exp1-DeepTrack-Accu.	0.58	0.61	0.51	0.86	0.54	0.35	0.73	0.49	0.54	0.61	0.81	0.66	0.51	0.72	0.76	0.60	0.62
Exp2-TPGR-Rob.	**0**	1.27	**0.20**	**0**	0.27	2.87	0.07	3.00	2.07	**0**	**0**	**0**	0.33	**0.07**	0.60	1.00	0.73
Exp2-DeepTrack-Rob.	0.27	**0**	0.33	**0**	**0.20**	**0.80**	**0**	**0.27**	**0.60**	**0**	**0**	**0**	**0**	0.07	**0.27**	**0.67**	**0.22**
Exp2-TPGR-Accu.	0.57	0.57	0.41	0.75	0.58	0.32	0.77	0.53	0.53	0.57	0.73	0.57	0.45	0.64	0.65	0.67	0.58
Exp2-DeepTrack-Accu.	0.54	0.62	0.49	0.77	0.50	0.36	0.70	0.47	0.53	0.59	0.75	0.62	0.60	0.69	0.69	0.56	0.59



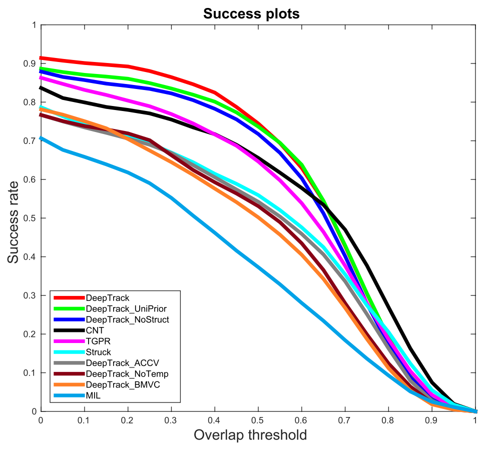


Fig. 10. The Precision Plot (left) and the Success Plot (right) of the results obtained by using different versions of DeepTrack. Note that the color of one curve is determined by the rank of the corresponding trackers, not their names.

label uncertainty and the structural loss is disabled and the yielded tracking results are shown in Fig. 10, compared with the full-version of the proposed method. In addition, the old versions of DeepTrack, *i.e.*, DeepTrack_BMVC [39] and DeepTrack_ACCV [36] is also compared in the Figure. Finally, the results of four well-known tracking methods, *i.e.*, MIL [19], Struck [4], TPGR [43] and CNT [42] are also shown as references.

From the figure we can see that, the structural loss, the temporal sampling mechanism and the label uncertainty all contribute the success of our CNN tracker. In particular, the temporal sampling plays a more important role. The structural loss can increase the TP accuracy by 10% and one can lifts the TP accuracy by 4% when the label noise is taken into consideration. Generally speaking, the curve consistently goes down when one component are removed from the original DeepTrack model. That indicates the validity of the proposed modifications.

E. Tracking Speed Analysis

We report the average speed (in fps) of the proposed DeepTrack method in Table V, compared with the DeepTrack without the truncated loss. Note that there are two kinds of average speed scores: the average fps over all the sequences and the average fps over all the frames. The latter one reduces the influence of short sequences where the initialization process usually dominates the computational burden.

TABLE V

THE TRACKING SPEED OF DEEPTRACK WITH OR WITHOUT THE TRUNCATED LOSS. NOTE THAT THERE ARE TWO KINDS OF AVERAGE SPEED SCORES: THE AVERAGE fps OVER ALL THE SEQUENCES (SEQUENCE AVERAGE) AND THE AVERAGE fps OVER ALL THE FRAMES (FRAME AVERAGE)

	Sequence Average	Frame Average
With TruncLoss	1.96fps	2.52fps
No TruncLoss	1.49fps	1.86fps

According to the table, the truncated loss boosts the tracking efficiency by around 37%. Furthermore, our method tracks the object at an average speed around 2.5fps. Considering that the speed of TPGR is around 3fps [43] and for the Sparse Representation based methods the speeds are usually lower than 2.5fps [25]. We thus can draw the conclusion that the DeepTrack can achieve comparable speed to the state-of-the-art methods.

F. Some Tracking Examples

In Fig. 11 we show the tracking results of our DeepTrack (magenta) comparing with other 5 state-of-the-art trackers, *i.e.*, TPGR (yellow), KCF (cyan), SCM (blue), Struck (red) and ASLA (black), on 12 challenging video sequences in the CVPR2013 benchmark. From row-1 to row-8 we show some good results obtained by using our algorithm. In row-9 and row-10 are the illustration that how the DeepTrack firstly lost

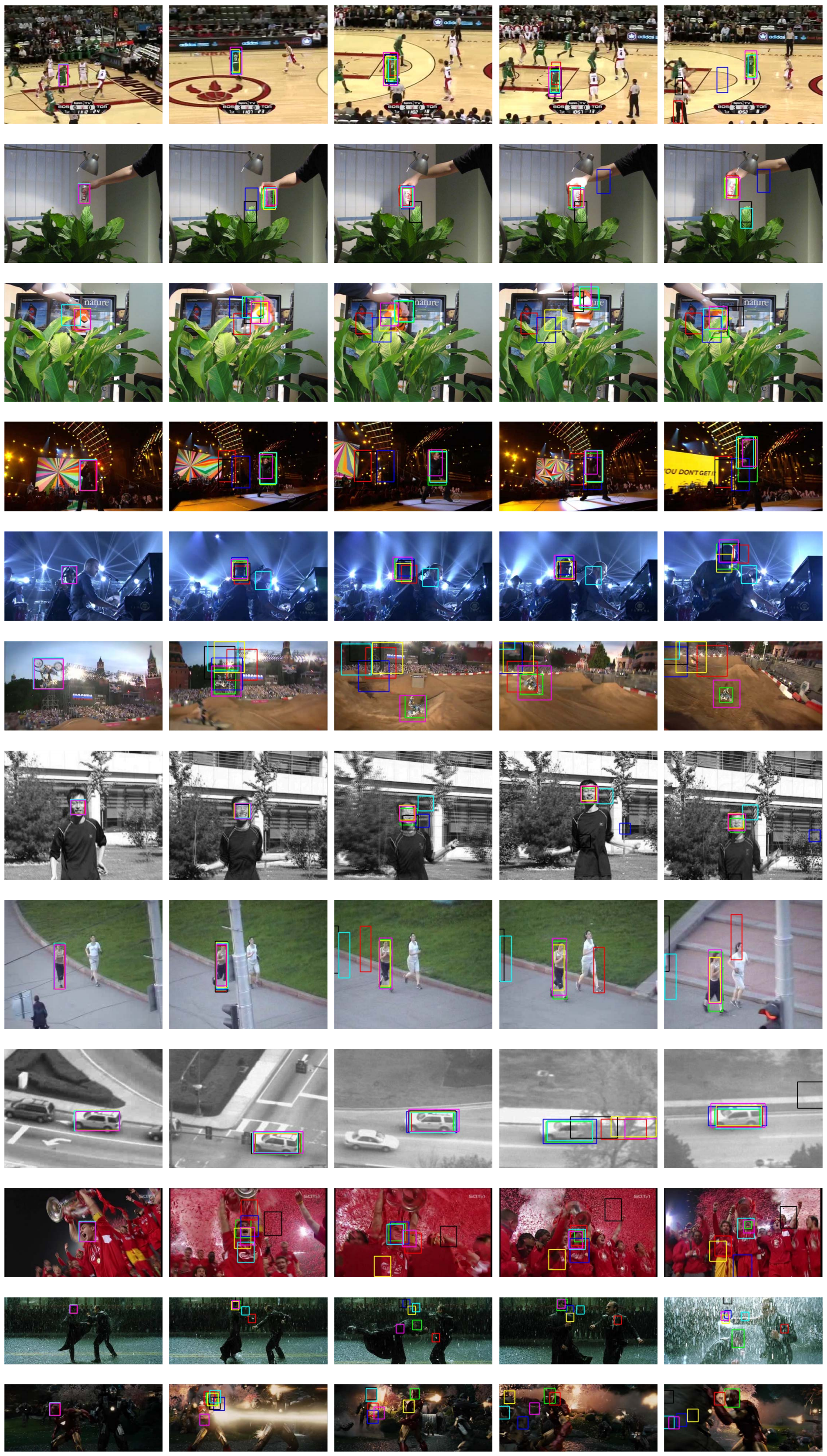


Fig. 11. Tracking results of the CNN tracker compared with other 6 visual trackers. The tracker are shown in different colors: green–ground-truth; red–Struck; blue–SCM; black–ASLA; yellow–TGPR; cyan–KCF; magenta–DeepTrack; In each row, the 5 frames roughly span uniformly over the corresponding sequence.

the target and then find it back after several frames, thanks to the robust temporal sampling scheme. The last two rows show the results on two difficult sequences (*m*atrix and *i*ronman), where DeepTrack achieves similar performance compared with other competitors. Note that on all the frames, the ground-truth bounding boxes are also plot in green.

V. Conclusion

We introduced a CNN based online object tracker. We employed a novel CNN architecture and a structural loss function to handle multiple input cues. We also proposed to modify the ordinary Stochastic Gradient Descent for visual tracking by iteratively update the parameters and add a robust temporal sampling mechanism in the mini-batch generation. This tracking- ed SGD algorithm increase the speed and the robustness of the training process significantly. Our experiments demonstrated that the CNN-based DeepTrack outperforms state-of-the-art methods on two recently proposed benchmarks which contain over 60 video sequences and achieves the comparable tracking speed.

References

[1] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Computer Vision—ECCV*. Heidelberg, Germany: Springer, 2002, pp. 661–675.

[2] 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.

[3] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. IEEE CVPR*, vol. 1. Jun. 2006, pp. 798–805.

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

[5] F. Yang, H. Lu, and M.-H. Yang, "Robust superpixel tracking," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1639–1651, Apr. 2014.

[6] B. Ma, L. Huang, J. Shen, and L. Shao, "Discriminative tracking using tensor pooling," *IEEE Trans. Cybern.*, vol. pp, no. 99, pp. 1–1, Sep. 2015.

[7] B. Ma, J. Shen, Y. Liu, H. Hu, L. Shao, and X. Li, "Visual tracking using strong classifier and structural local sparse descriptors," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1818–1828, Oct. 2015.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1. Jun. 2005, pp. 886–893.

[10] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[12] K. Kavukcuoglu, P. Sermanet, Y-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, "Learning convolutional feature hierarchies for visual recognition," in *Proc. NIPS*, 2010, pp. 1090–1098.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[14] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. CVPR*, Jun. 2012, pp. 3642–3649.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.

[16] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNS for fine-grained category detection," in *Computer Vision—ECCV*. Springer International Publishing, 2014, pp. 834–849.

[17] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1610–1623, Oct. 2010.

[18] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. NIPS*, 2013, pp. 809–817.

[19] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 983–990.

[20] N. D. Lawrence and B. Schölkopf, "Estimating a kernel fisher discriminant in the presence of label noise," in *Proc. ICML*, 2001, pp. 306–313.

[21] P. M. Long and R. A. Servedio, "Random classification noise defeats all convex potential boosters," *Mach. Learn.*, vol. 78, no. 3, pp. 287–304, 2010.

[22] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, Inc., 2013, pp. 1196–1204.

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

[24] R. B. Girshick, P. F. Felzenszwalb, and D. A. McAllester, "Object detection with grammar models," in *Advances in Neural Information Processing Systems*. Neural Information Processing Systems Foundation, Inc., 2011, pp. 442–450.

[25] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, "Robust object tracking with online multi-lifespan dictionary learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 665–672.

[26] O. Matan, C. J. C. Burges, Y. Le Cun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *Neural Information Processing Systems*. San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 488–495.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision—ECCV*. Springer International Publishing, 2014, pp. 346–361.

[28] R. Girshick. (2015). "Fast R-CNN." [Online]. Available: http://arxiv.org/abs/1504.08083

[29] S. Ren, K. He, R. Girshick, and J. Sun. (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks." [Online]. Available: http://arxiv.org/abs/1506.01497

[30] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

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

[32] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–12.

[33] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[34] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *Computer Vision*. Springer, 2008, pp. 2–15.

[35] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comp. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[36] H. Li, Y. Li, and F. Porikli, "Robust online visual tracking with a single convolutional neural network," in *Proc. 12th ACCV*, 2014, pp. 194–209.

[37] 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.

[38] P. Viola, J. C. Platt, and C. Zhang, "Multiple instance boosting for object detection," in *Proc. NIPS*, 2005, pp. 1417–1424.

[39] H. Li, Y. Li, and F. Porikli, "DeepTrack: Learning discriminative feature representations by convolutional neural networks for visual tracking," in *Proc. BMVC*, 2014, pp. 1–12.

[40] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. CVPR*, 2013, pp. 2411–2418.

[41] M. Kristan *et al.*, "The visual object tracking VOT2013 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Dec. 2013, pp. 98–111.

[42] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang. (2015). "Robust visual tracking via convolutional networks." [Online]. Available: http://arxiv.org/abs/1501.04505

[43] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Computer Vision—ECCV*. Springer International Publishing, 2014, pp. 188–203.

[44] 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.

[45] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints," in *Proc. IEEE CVPR*, Jun. 2010, pp. 49–56.

[46] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. IEEE CVPR*, Jun. 2010, pp. 1269–1276.

[47] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *Proc. IEEE CVPR*, Jun. 2011, pp. 1177–1184.

[48] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE CVPR*, Jun. 2012, pp. 1822–1829.

[49] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE CVPR*, Jun. 2012, pp. 1838–1845.

[50] A. Wendel, S. Sternig, and M. Godec, "Robustifying the flock of trackers," in *Proc. 16th Comput. Vis. Winter Workshop*, 2011, p. 91.

[51] M. Felsberg, "Enhanced distribution field tracking using channel representations," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Dec. 2013, pp. 121–128.

[52] J. Xiao, R. Stolkin, and A. Leonardis, "An enhanced adaptive coupled-layer LGTracker++," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Dec. 2013, pp. 137–144.

[53] T. Nawaz and A. Cavallaro, "A protocol for evaluating video trackers under real-world conditions," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1354–1361, Apr. 2013.

Hanxi Li received the Ph.D. degree from the Research School of Information Science and Engineering, Australian National University, Canberra, Australia. He was a Researcher with NICTA (Australia) from 2011 to 2015. He is a Special Term Professor with the School of Computer and Information Engineering, Jiangxi Normal University, China. His recent areas of interest include visual tracking, face recognition, and deep learning.

Yi Li received the Ph.D. degree from the ECE Department, University of Maryland, College Park. He was a Senior Researcher with NICTA (Australia) from 2011 to 2015. His Ph.D. research, entitled Cognitive Robots for Social Intelligence, focuses on visual navigation for mobile robots, optical motion capture, causal inference for coordinated groups, and action recognition and representation. He was the Recipient Future Faculty Fellow at the University of Maryland from 2008 to 2010. He is a Senior Research Scientist with the Toyota Research Institute, Ann Arbor, MI, and an Adjunct Fellow with Australian National University, Canberra, Australia. His recent areas of interests include autonomous driving, robotics, and deep learning. He received the Best Student Paper of ICHFR, and the second price in the Semantic Robot Vision Challenge. He co-organized the first two DeepVision workshops in the IEEE Conference on Computer Vision and Pattern Recognition, and served as the Area Chair of the IEEE Winter Conference on Applications of Computer Vision in 2015 and 2016.

Fatih Porikli (F'14) received the Ph.D. degree from the New York University, NY. He is currently a Professor in the Research School of Engineering, Australian National University (ANU). He is also managing the Computer Vision Research Group at Data61. Until 2013, he was a Distinguished Research Scientist with Mitsubishi Electric Research Labs (MERL), Cambridge, USA. His research interests include computer vision, pattern recognition, manifold learning, sparse optimization, online learning, and image enhancement with commercial applications in video surveillance, intelligent transportation, satellite, and medical systems. He authored more than 140 publications and invented 66 patents. He received the R&D100 2006 Award in the Scientist of the Year category in addition to four IEEE Best Paper Awards and five Professional Prizes. He serves as an Associate Editor of the *IEEE Signal Processing Magazine*, the *SIAM Journal on Imaging Sciences RealTime Image and Video Processing* (Springer), and the *EURASIP Journal on Image and Video Processing*. He was the General Chair of the IEEE Winter Conference on Applications of Computer Vision in 2014 and the IEEE Advanced Video and Signal Based Surveillance Conference in 2010, and serves at the Organizing Committee of many IEEE events.