Applied Soft Computing 38 (2016) 1088–1098

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

CNNTracker: Online discriminative object tracking via deep convolutional neural network

Yan Chen*, Xiangnan Yang, Bineng Zhong, Shengnan Pan, Duansheng Chen, Huizhen Zhang

Department of Computer Science and Technology, Huaqiao University, China

ARTICLE INFO

Article history:
Received 28 November 2014
Received in revised form 23 June 2015
Accepted 28 June 2015
Available online 6 July 2015

Keywords:
Deep learning
Object tracking
Convolutional neural network
Object appearance model
Large scale training data

ABSTRACT

Object appearance model is a crucial module for object tracking and numerous schemes have been developed for object representation with impressive performance. Traditionally, the features used in such object appearance models are predefined in a handcrafted offline way but not tuned for the tracked object. In this paper, we propose a deep learning architecture to learn the most discriminative features dynamically via a convolutional neural network (CNN). In particular, we propose to enhance the discriminative ability of the appearance model in three-fold. First, we design a simple yet effective method to transfer the features learned from CNNs on the source tasks with large scale training data to the new tracking tasks with limited training data. Second, to alleviate the tracker drifting problem caused by model update, we exploit both the ground truth appearance information of the object labeled in the initial frames and the image observations obtained online. Finally, a heuristic schema is used to judge whether updating the object appearance models or not. Extensive experiments on challenging video sequences from the CVPR2013 tracking benchmark validate the robustness and effectiveness of the proposed tracking method.

1. Introduction

Object tracking has attracted substantial attention due to its potential value both in theoretical challenges as well as in practical applications, including intelligence video surveillance, self-driving vehicles, robotics and so on. Despite much progress has been made in recent years, designing robust object tracking methods is still a challenging problem. In the core of such challenges lies the difficulty of robust appearance modeling, with respect to the object appearance variations caused by illumination changes, occlusions, pose changes, cluttered scenes, moving backgrounds, etc. To this end, most the existing methods focus on the two aspects of building a robust object appearance model, i.e., feature representation and classifier construction.

Feature representation: up to now, a variety of well-known features have been introduced for object tracking, including color histograms [1] or attributes [2], subspace-based features [3,4], Haar-like features [5–8], LBP [9–11], HoG [12–14], SFIT [15,16], SURF [17], covariance matrix [18,19], 3D-DCT [20], shape features [21], and combining several complementary cues, etc. While these predefined and handcrafted features have achieved great success for some specific data and tasks, they are low-level features and not tuned for the tracked object. Moreover, to capture the object appearance variations during tracking, designing effective features for object tracking usually requires reflecting the time-varying properties of object appearance model. However, most handcrafted features cannot be simply adapted according to the new observed data. Thus, online learning features from the object (or data) of interest is considered as a plausible way to remedy the limitation of handcrafted features.

Classifier construction: besides the feature representation, classifier construction is another important problem in designing a robust object appearance model. The typical classifiers include SVM [22,23], boosting [6,24], random forest [25], Hough forest [26], structural learning [7,8,27], sparse coding [28–33], discriminative feature learning [34], multiple instance learning [35], co-training technique [36], tracking-learning-detection [37], weakly supervised learning [38,39], and-or graphs [14], coupled 2-layer model [40], etc. However, most of these classifiers are limited by their shallow or linear nature structures while object appearance variations are complex, highly nonlinear, and time-varying.

* Corresponding author at: Jimei District, Huaqiao University, Xiamen, Fujian 361021, China. Tel.: +86 592 6162556.
E-mail address: yannychen@hqu.edu.cn (Y. Chen).

http://dx.doi.org/10.1016/j.asoc.2015.06.048
1568-4946/

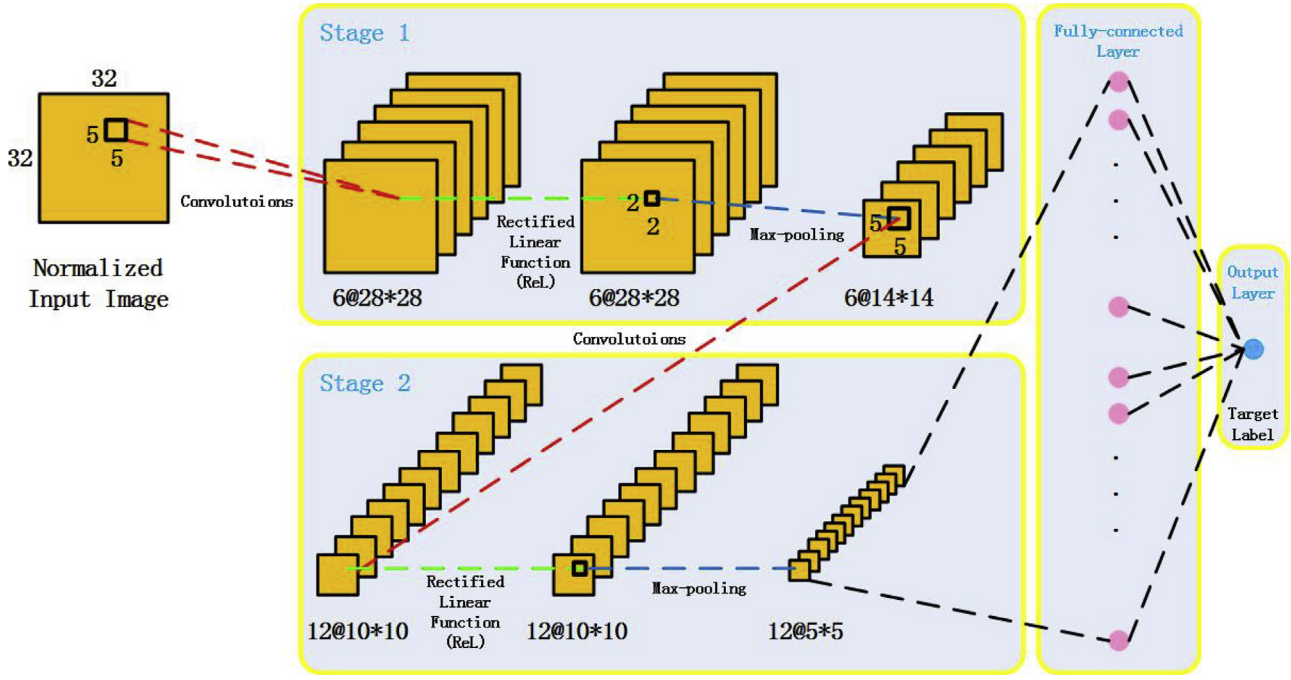

Fig. 1. Illustration of how the proposed CNNTracker construct an object appearance model from a deep convolutional neural network. The raw input image is fed to a 2-stage convolutional neural network, in which each stage contains a filter bank layer, a rectified linear layer, and a max-pooling layer respectively. Best viewed in color.

In this work, inspired by the success of deep learning [41–47], we propose an object tracking method (termed CNNTracker) that relies on deep convolutional neural network to address both limitations of handcrafted features and shallow classifier structures in object tracking problem. As shown in Fig. 1, the main idea behind our CNNTracker is to use a convolutional neural network trained from raw pixels to produce a discriminative appearance model. More specifically, the discriminative features are first automatically learned via a deep convolutional neural network, which is composed of multiple stages, each of which contains a filter bank layer, a nonlinearity layer, and a spatial pooling layer respectively. With end-to-end training, CNNTracker can alleviate the need for handcrafted features, and automatically learn hierarchical and object-specific feature representations. Second, to alleviate the tracker drifting problem caused by model update, we exploit both the ground truth appearance information of the object labeled in the initial frames and the image observations obtained online. Finally, a heuristic schema is used to judge whether updating the object appearance models or not.

The advantages of our CNNTracker are three-fold:

(1) Our CNNTracker is online trained with a multi-layer deep model in a supervised manner, and has more discriminative and representative power than the previous tracking methods using handcrafted features and shallow models.
(2) Our CNNTracker effectively exploits both the ground truth appearance information of the object labeled in the initial frames and the image observations obtained online, and thus can alleviate the tracker drift problem.
(3) Our CNNTracker can use a heuristic schema to judge whether updating the object appearance models or not.

The rest of the paper is organized as follows. An overview of the related work is given in Section 2. Section 3 introduces how to learn a deep CNN-based object appearance model. The detailed tracking method is then described in Section 4. Experimental results are given in Section 5. Finally, we conclude this work in Section 6.

2. Related work

There has been a significant amount of progress made in the area of object tracking. Please refer to the survey papers [48–50] and a recent benchmark [51] for a comprehensive survey of this topic. Since our work focuses on utilizing deep learning to develop effective object appearance models for robust object tracking. We first review recent online tracking methods using different types of object appearance models, namely, generative and discriminative methods. Moreover, some tracking methods based on multi-cue fusion and part-based models are also briefly reviewed. Then, we discuss deep learning and its applications in tracking communities.

2.1. Online object tracking

One of the key challenges of object tracking is to develop effective object appearance models for handling illumination changes, occlusions, pose changes, cluttered scenes, moving backgrounds, etc. Recently, numerous object representation strategies have been proposed, which focus on constructing online object appearance models to account for the inevitable appearance changes of objects. Typically, the online tracking methods using different types of object appearance models can be classified into two categories: generative and discriminative methods.

Generative tracking methods use some generative process to describe the object appearance models and make a decision based on the reconstruction errors or the matching scores. Popular generative methods include kernel-based tracker [1], Gaussian mixture model-based tracker [52], subspace-based tracker [3], covariance-based tracker [18], low-rank and sparse representation-based trackers [28–33], and visual tracking decomposition [53], and so on. Specifically, Comaniciu et al. [1] propose a kernel-based tracking method using color histogram-based object representations. To handle object appearance changes, Jepson et al. [52] use an online EM algorithm to construct an adaptive Gaussian mixture model-based object representation. In [3], an incremental principal component analysis (PCA)-based tracking method is

proposed. In [18], Porikli et al. propose a tracking method using a covariance based object description and a Lie algebra based update mechanism. Recently, a variety of low-rank subspaces and sparse representations based tracking methods have been proposed [28–33] for object tracking due to their robustness to occlusion and image noises. Mei et al. [28] are among the first to utilize the technique and propose an L1 minimization-based tracking method, in which the target is represented by a sparse linear combination of object templates and trivial templates. However, it is time consuming. To improve the time complexity, many extensions [29,30] have also been proposed. Zhang et al. [31] exploit the relationship between particles via low rank sparse learning for object tracking. In [32], Jia et al. propose a tracking method based on the structural local sparse appearance model. Zhang et al. [33] propose a tracking method relying on mean shift, sparse coding and spatial pyramids. Kwon et al. [53] propose an object tracking decomposition method, which decomposes an observation model into multiple basic observation models to capture a wide range of pose and illumination changes. While generative tracking methods usually produce more accurate results under less complex environments due to the richer image representations used, they are prone to drift in more complex environments without utilizing the target and background information.

Unlike generative tracking methods, discriminative tracking methods learn to explicitly distinguish the target object from its surrounding background by formulating tracking as a binary classification problem. Based on the variance ratio of two classes, Collins et al. [34] select discriminative color features for constructing the adaptive appearance models. Avidan [54] uses an adaptive ensemble of weak classifiers for object tracking. Grabner et al. [6] propose an online boosting based tracking algorithm. Unfortunately, one inherent problem of online learning-based trackers is drift, a gradual adaptation of the tracker to non-targets. To alleviating the drifting problem, Matthews et al. [55] provide a partial solution for template trackers by assuming the current tracker does not stray too far from the initial appearance model. Combining with a prior classifier, Grabner et al. [24] propose a semi-supervised online boosting algorithm for object tracking. However, the method cannot accommodate very large changes in appearance due to using the prior classifier. Babenko et al. [35] propose a tracking method using a multiple instance boosting based appearance model. Hare et al. [7] design a tracking system based on an online kernelized structured output support vector machine.

Some authors attempt to utilize multiple cues information to improve the performance of a tracker. Santner et al. [25] propose a tracking method, which combines an online random forest-based tracker, a correlation-based template tracker, and an optical-flow-based mean shift tracker in a cascade-style. Zhang and Maaten [27] propose a multi-object model-free tracker by using an online structured SVM algorithm to learn the spatial constraints along with the object detectors. Duffner and Garcia [36] present a method for tracking non-rigid objects, which combines and adapts a detector and a probabilistic segmentation method in a co-training manner. The co-training based algorithms require the independence among different features, which is too strong to be met in practice. Kalal et al. [37] propose a TLD-based tracking system that explicitly decomposes the long-term tracking task into tracking, learning, and detection. Zhong et al. [38] propose a weakly supervised learning based tracking method to fuse several complementary trackers. Kwon and Lee [57] propose a visual tracker sampler, in which multiple appearance models, motion models, state representation types, and observation types are sampled via Markov Chain Monte Carlo to generate the sampled trackers. Zhang et al. [58] propose a tracking method via multiple experts using entropy minimization. Wang and Yeung [59] propose an ensemble-based tracking method via aggregating crowdsourced structured time series data. However, most of these methods use a single global bounding box to delineate the entire target object. Consequently, they may be sensitive to partial occlusion and pose variation.

To explicitly handle the occlusion problem, Yao et al. [8] propose a part-based object tracking method with online latent structural learning. Gall et al. [26] propose a Hough forests-based tracking method. Cehovin et al. [40] propose a coupled-layer based tracking method that combines the target's global and local appearance by interlacing two layers. Zhang et al. [56] propose a part matching based tracking method via locality-constrained low-rank sparse learning. Although part-based tracking methods have been successful on dealing with the occlusion problem, they may be limited by the choice of the low-level features applied to the image patches.

2.2. Deep learning

Recently, deep learning methods, together with the increased processing power and the advances in graphics processors, have come to play a vital role in big data analytics and several tasks, from speech recognition [42] to image classification [43], pedestrian detection [46], generic visual recognition [47], face recognition [60], and image super-resolution [61]. Some companies (e.g., Google [67] and Facebook [68]) have also launched deep learning related projects via collecting and analyzing massive amounts of data on a daily basis.

Deep learning is a set of algorithms in machine learning that attempt to model high-level abstractions in data by using model architectures composed of multiple non-linear transformations [44]. Representative deep learning architectures for vision include the deep belief networks [62] and convolutional neural networks [43] et al.

A deep belief network is a generative graphical model composed of a layer of visible units and multiple layers of hidden units, with connections between the layers but not between units within each layer. The deep belief networks and related unsupervised learning algorithms such as autoencoders and sparse coding have been used to learn higher-level feature representations from unlabeled data, and produced state-of-the-art results on recognition and classification tasks [44].

A convolutional neural network is a variant of feed-forward artificial neural network and based on two key concepts: local receptive fields, and weight-tying. With the availability of large scale datasets like ImageNet and high computational power with GPUs, the convolutional neural networks have achieved impressive results on image recognition and object detection [43,45–47].

Although deep learning methods have shown top-performing results in many tasks, their training is time-consuming and not a trivial task for big data analytics. Consequently, with the explosion of data sets in recent years, some researchers have developed effective and scalable parallel algorithms for training large-scale deep learning-based models [69]. On the other hand, some researchers use a high-end GPU or multiple GPUs/clusters to speed up the training process of deep learning from big data [70–73]. Raina et al. [70] proposed a GPU-based framework for massively parallelizing deep belief networks. Moreover, there have been a few studies on approximating deep models for accelerating test-time evaluation [74,75].

In spite of the recent success in large-scale deep learning, how to address typical challenges posted by big data is still an open issue, e.g., large scale of data, different types of data, and the speed of streaming data, respectively [76,77].

In this paper, we focus on how to construct an effective deep CNN based appearance model for discriminative object tracking. There have been a few of earlier work in applying deep learning to object tracking [63–65]. Carneiro and Nascimento [63] propose a tracking method for the left ventricle endocardium in ultrasound

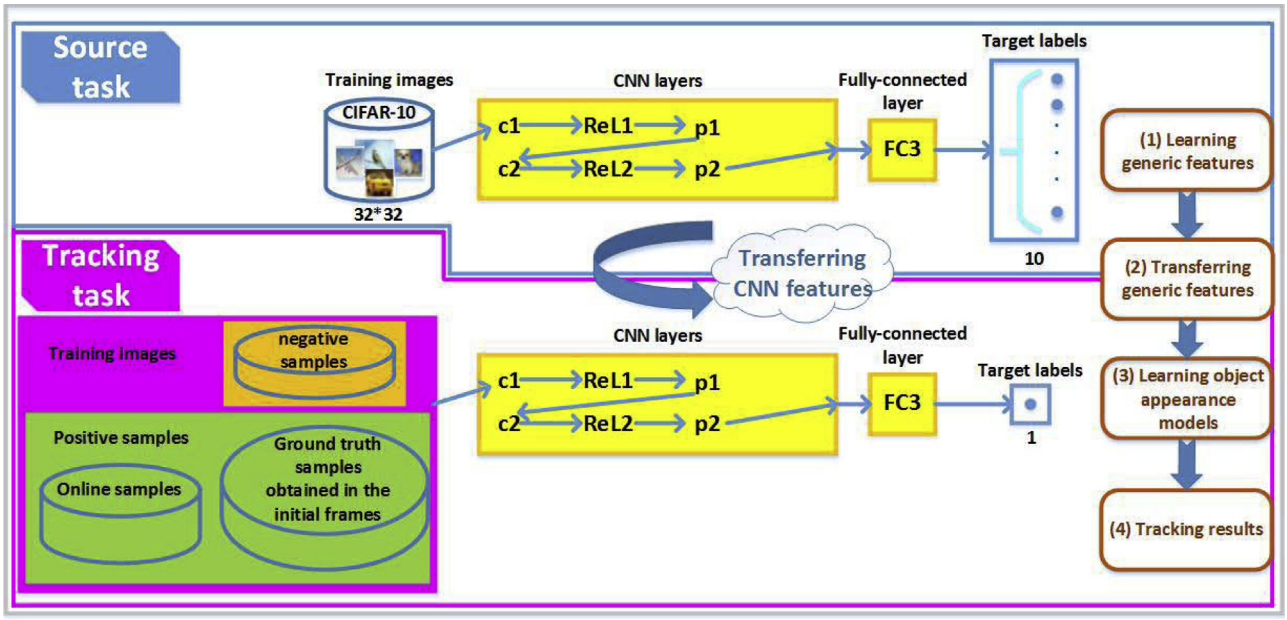

Fig. 2. Learning object appearance models by transferring the CNN features. First, the deep CNN is pre-trained on the source task (CIFAR-10 classification, top row). Then, the pre-trained parameters of the internal layers of the CNN (c1-FC3) are then transferred to the tracking task (bottom row). To achieve the transfer and construct the object appearance models, we remove the output layer with 10 units and add an output layer with one unit. Furthermore, to alleviate the drifting problem, we exploit both the initial and online samples to update the object appearance models. Best viewed in color.

data, which combines multiple dynamic models and deep learning architectures. Wang and Yeung [64] propose a stacked denoising autoencoder based tracking method, which firstly learns generic image features from an offline dataset and then transfers the features learned to the online tracking task. Fan et al. [65] propose a human tracking method via a convolutional neural network, in which the features are learned during offline training.

3. Object appearance model

Convolutional neural network [43] is a biologically inspired class of deep learning models that has achieved excellent performance on visual and speech recognition problems. In this section, we address the problem of how to learn a data-driven object appearance model from a deep convolutional neural network. The deep convolutional neural network is fed with raw image pixels and trained in supervised mode from training samples to produce a likelihood evaluation for each candidate sample.

3.1. Deep convolutional neural network

Deep convolutional neural networks (CNN) offer a class of hierarchical models to learn features directly from image pixels. We train our deep CNN on an object tracking task. The overall architecture of our deep CNN is shown in Fig. 1. The raw input image is fed to a 2-stage convolutional neural network, in which each stage contains a filter bank layer, a rectified linear layer, and a max-pooling layer respectively.

More specifically, in the first stage, the input is a pre-processed gray image of size 32 by 32 pixels and is given to a convolutional layer with 6 filters of size 5×5. The resulting 6 feature maps of size 28 by 28 pixels are then fed to a non-linear transformation layer which applies the following point-wise rectified linear unit (ReLU) transformation to the whole feature maps:

$$\mathrm{relu}(x) = \max(0, x) \tag{1}$$

The resulting 6 feature maps are then further fed to a max-pooling layer, which takes the max over 2×2 spatial neighborhoods. Deep networks can be trained efficiently using ReLU without pre-training. Also, ReLU does not face gradient vanishing problem as with sigmoid and tanh function. Max-pooling layers make the output of convolution networks more robust to small translations and geometric distortions. When applied to object tracking, they make the network more robust to small tracking errors caused by slight inaccuracies/offsets in the object location.

Similarly, the second stage consists of (i) convolution of the first stage output; (ii) passing the responses through a ReLU function; and (iii) max pooling over local 2×2 neighborhoods. After the above two stages of convolutions, ReLU transformation, and max-pooling, the network can extract high level features.

3.2. Learning object appearance models from deep CNN

In this paper, object tracking is formulated as an online transfer learning problem and the deep CNN is used to construct the object appearance model due to its powerful capacity for automatically learning a hierarchical feature representation. As shown in Fig. 2, the key idea is to use the internal CNN features as a generic and middle level image representation, which can be pre-trained on one dataset (the source task, here CIFAR-10 [66]) and then re-used (fine-tuned) on the tracking tasks.

More specifically, for the source task, we pre-train a deep CNN with two CNN layers followed by one fully-connected layer from the CIFAR-10 natural image dataset [66]. The CIFAR-10 dataset is a labeled subset of the 80 million tiny images, containing 60,000 images and 10 classes. Each CNN layer is composed of a convolutional, ReLU and pooling layer. The output layer has size 10 equal to the number of target categories.

After pre-training on the source task, the parameters of layers c1, ReL1, p1, c2, ReL2, p2, and FC3 are first transferred to the tracking task. Then, we remove the output layer with ten units and add an output layer with one unit. Finally, the newly designed CNN is re-trained on the training data from the tracking task to learn an object appearance model. This simple yet effective transferring schema

Algorithm 1 Object Tracking via the Deep CNN-based Appearance Model

Initialization:

1. Pre-train a deep CNN on the CIFAR-10 dataset.
2. Acquire one manually labels the first frame.
3. Collect positive samples $s_1^+ = \{x_{1,i}^+\}_{i=1}^{N_1^+}$, negative samples $s_1^- = \{x_{1,i}^-\}_{i=1}^{N_1^-}$, and crop out the corresponding image patches.
4. Resize each positive/negative image patch to 32*32 pixels.
5. Fine-tune the pre-trained deep CNN-based appearance model based on s_1^+ and s_1^-.
6. Initialize the particle set $\{x_1^i, w_1^i\}_{i=1}^{N_1}$ at time t=1, where $w_1^i = 1/N_1, i = 1,...,N_1$
7. Set the maximum buffer size T_1 for the set of online positive samples s_{ot}^+.
8. Set the likelihood threshold T_2.

for t = 2 to the end of the video

1. Prediction: for $i = 1,...,N_1$, generate $x_t^i \sim p(x_t \mid x_{t-1}^i)$
2. Likelihood evaluation: for $i = 1,...,N_1$, let $w_t^i = w_{t-1}^i p(y_t \mid x_t^i)$.
3. Determine the optimal object state x_t^* as the particle with the maximum weight.
4. Resample: Normalize the weights and compute the covariance of the normalized weights. If this variance exceeds one threshold, then $\beta_j \sim \{w_t^i\}_{i=1}^{N_1}$ and replace $\{x_t^i, w_t^i\}_{i=1}^{N_1}$ with $\{x_t^{\beta_j}, 1/N_1\}_{j=1}^{N_1}$
5. If the posterior density of the optimal object state $p(x_t^* \mid y_{1:t})$ is within [T_2, T_3], then,
   - 5.1 Select positive and negative samples $s_t^+ = \{x_{t,i}^+\}_{i=1}^{N_t^+}$ and $s_t^- = \{x_{t,i}^-\}_{i=1}^{N_t^-}$ at time t respectively.
   - 5.2 Update the set of online positive samples $s_{ot}^+ = s_{ot}^+ \bigcup s_t^+$.
   - 5.3 If the size of s_{ot}^+ is larger than T_1, then s_{ot}^+ is truncated to keep the last T_1 elements.
   - 5.4 Update the final positive sample sets $s^+ = s_1^+ \bigcup s_{ot}^+$.
   - 5.5 Update the deep CNN-based appearance model based on s^+ and s_t^-.

end for

Fig. 3. Summary of the proposed CNNTracker.

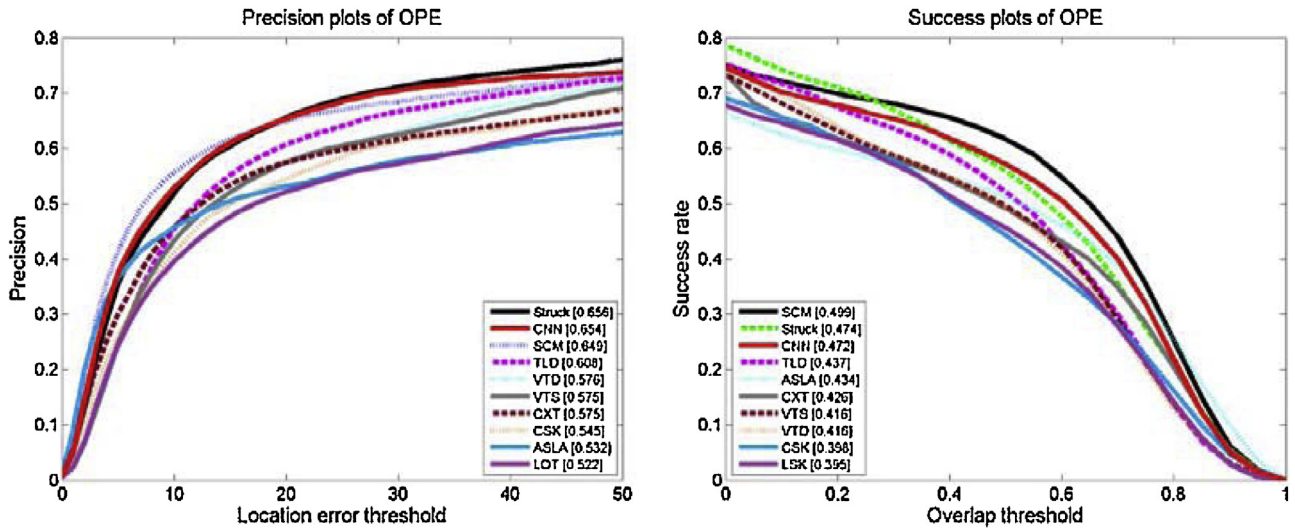

Fig. 4. The precision and success plots of quantitative comparison for the 50 sequences in the CVPR2013 tracking benchmark [51]. The performance score of each tracker is shown in the legend. The proposed CNNTracker (in red) obtains better or comparable performance against state-of-the-art tracking methods. (For interpretation of the references to color in text, the reader is referred to the web version of this article.) .

enables the proposed CNNTracker to tackle the domain changes in training tasks.

4. Object tracking via deep convolutional neural network (CNNTracer)

In this section, we develop the proposed tracking method based on the appearance model described above. The basic idea is to efficiently incorporate the deep CNN-based appearance model into the particle filtering framework, which implements recursive Bayesian filter by Monte Carlo sampling. The main idea behind particle filter is to represent the posterior density by a set of random particles with associated weights. There are two main components in particle filter:

(1) Dynamic model: generate candidate samples based on previous particles.
(2) Observation model: calculate the similarity between candidate samples and the target appearance model.

Given all observations of the object $y_{1:t} = [y_1, \ldots, y_t]$ up to time t, the aim of a particle filter-based tracking system is to estimate $p(x_t|y_{1:t})$, which is a posterior density of target state. Using the Bayesian theorem, the posterior density $p(x_t|y_{1:t})$ can be reformulated:

$$p(x_t|y_{1:t}) \propto p(y_t|x_t) \int p(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}) dx_{t-1} \tag{2}$$

where $p(x_t|x_{t-1})$ is the dynamic model and $p(y_t|x_t)$ is the observation model. Calculation of the integral is carried out by Monte Carlo sampling in particle filter. In other words, the posterior density $p(x_t|y_{1:t})$ is approximated by a set of samples (particles) $\{x_t^i\}_{i=1}^{N_t}$ with associated weights $\{w_t^i\}_{i=1}^{N_t}$.

Finally, the optimal object state x_t^* at time t can be determined by the maximum a posterior estimation:

$$x_t^* = arg \quad \max_{x_t} p(x_t|y_{1:t}) = x_t^i = arg \quad \max_{x_t^i} w_t^i \tag{3}$$

In this paper, for simplicity and computational efficiency reasons, we choose to track only the location and size. Specifically,

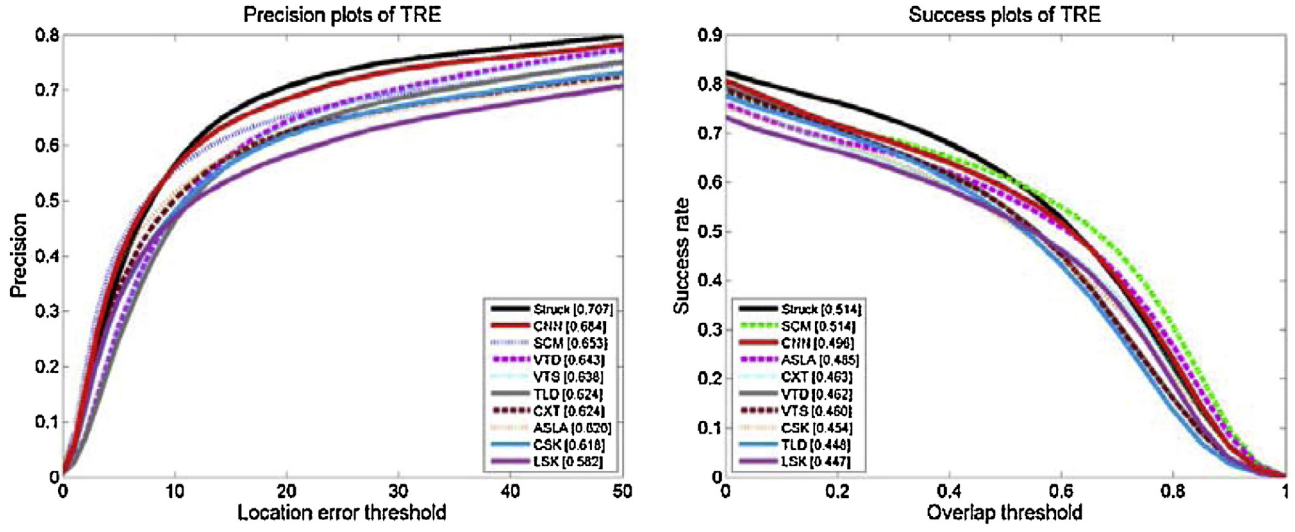

Fig. 5. The precision and success plots for TRE. The proposed CNNTracker (in red) achieve comparable performance in all the evaluations. (For interpretation of the references to color in text, the reader is referred to the web version of this article.)

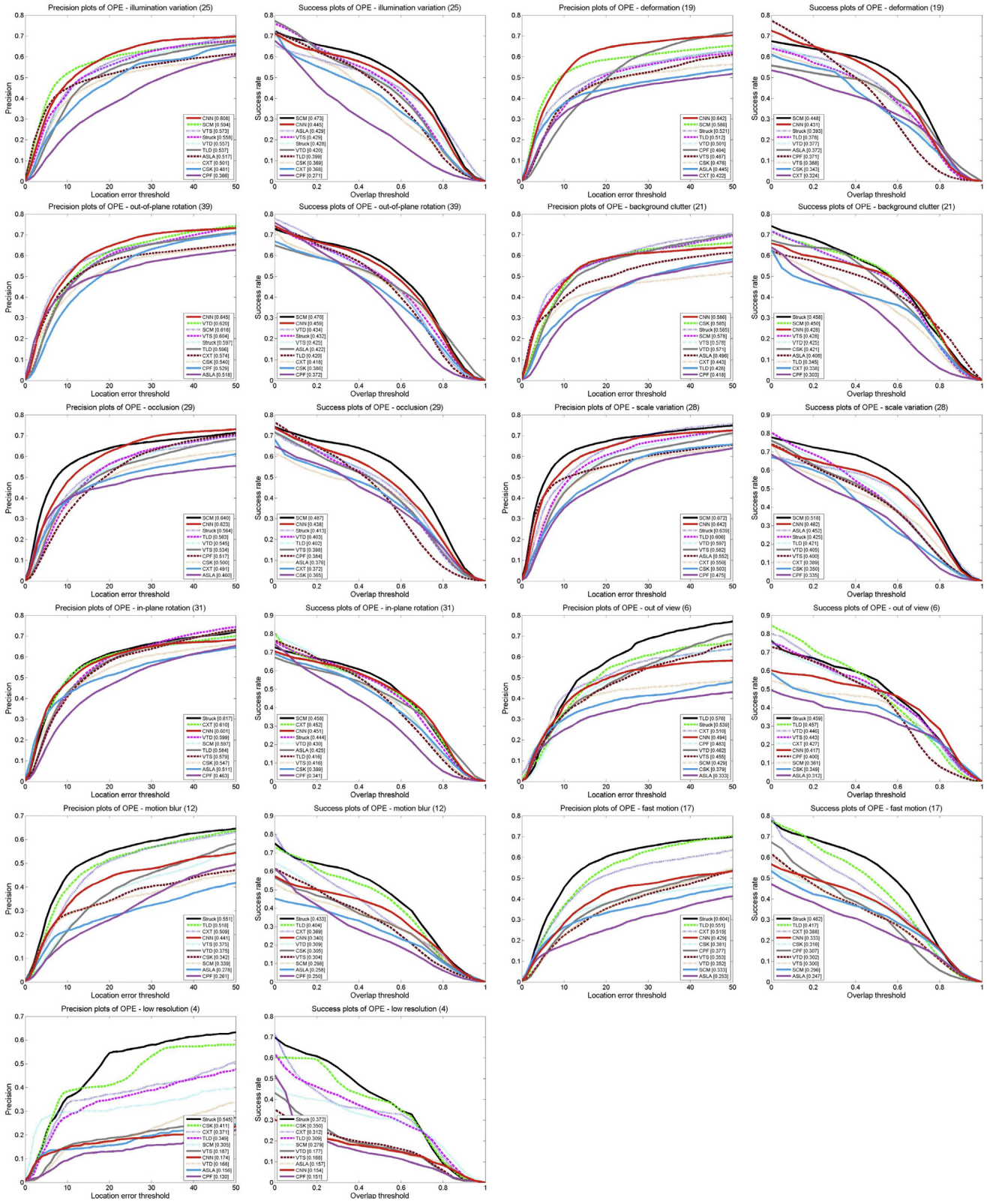

Fig. 6. Plots for different subsets divided based on main variation of the target object. The value appearing the title denotes the number of videos tagged with the main variation. The proposed CNNTracker (in red) achieves comparable performance against state-of-the-art trackers (For interpretation of the references to color in text, the reader is referred to the web version of this article.).

Fig. 7. Qualitative comparison on several sequences from [51], i.e., the doll, freeman 4, jogging-2, and trellis sequence respecitvely.

let $x_t = (p_t^x, p_t^y, w_t, h_t)$ denote the object state parameters including the horizontal coordinate, vertical coordinate, width and height respectively. The dynamic model between two consecutive frames is assumed to be a Gaussian distribution:

$$p(x_t|x_{t-1}) = N(x_t; x_{t-1}, \sum) \tag{4}$$

where $\sum$ is a diagonal covariance matrix whose diagonal elements are the corresponding variances of respective parameters. For each state x_t, there is a corresponding image patch that is normalized to 32×32 pixels by image scaling. The likelihood function $p(y_t|x_t)$ is calculated based on the proposed deep CNN-based appearance model. With an output score from the output layer of the deep CNN-based appearance model d_t, the likelihood function is calculated by:

$$p(y_t|x_t) = \exp(d_t) \tag{5}$$

To capture the appearance variations, the likelihood function needs to adapt over time due to updating the deep CNN-based appearance model. However, the main drawback of the appearance-adaptive methods is their sensitivity to drift, i.e., they may gradually adapt to non-targets. To alleviate the drifting problem, we exploit both the ground truth appearance information of the object labeled in the initial frames and the image observations obtained online. Specifically, we assume the positive samples obtained in the first frame to be $s_1^+ = \{x_{1,i}^+\}_{i=1}^{N_1^+}$. The online positive samples obtained in the most recent frames are denoted as $s_{ot}^+ = \{x_{t-i}^+\}_{i=1}^{T_1}$. The positive and negative samples obtained at current frame t are denoted as $s_t^+ = \{x_{t,i}^+\}_{i=1}^{N_t^+}$ and $s_t^- = \{x_{t,i}^-\}_{i=1}^{N_t^-}$ respectively. At current frame t, if the posterior density of the optimal object state $p(x_t^*|y_{1:t})$ is smaller than a predefined threshold T_2 or larger than a predefined threshold T_3, we do not update the deep CNN-based appearance model. Otherwise, based on s_1^+, s_{ot}^+, s_t^+, and s_t^-, we update the objet appearance model.

The basic idea behind this heuristic schema is that if the likelihood is smaller than a predefined threshold T_2, the current tracking results may be unreliable. Thus we should not update the object appearance model. Meanwhile, if the likelihood is larger than a predefined threshold T_3, the current tracking results is reliable. Thus, we should believe the tracking results and not update the object appearance model. Otherwise, we should update the object appearance model to capture the gradually object appearance variations.

Finally, a summary of our CNN-based tracking method is described in Fig. 3.

5. Experiments

In this section, we first introduce the setting of our experiments. Then, we apply the proposed CNNTracker to the 50 challenging sequences from the CVPR2013 tracking benchmark [51], and systematically compare it with the 30 state-of-the-art trackers.

5.1. Experiment setting

The proposed CNNTracker is implemented in Matlab on a HP Z800 workstation with an Intel(R) Xeon(R) E5620 2.40 GHz

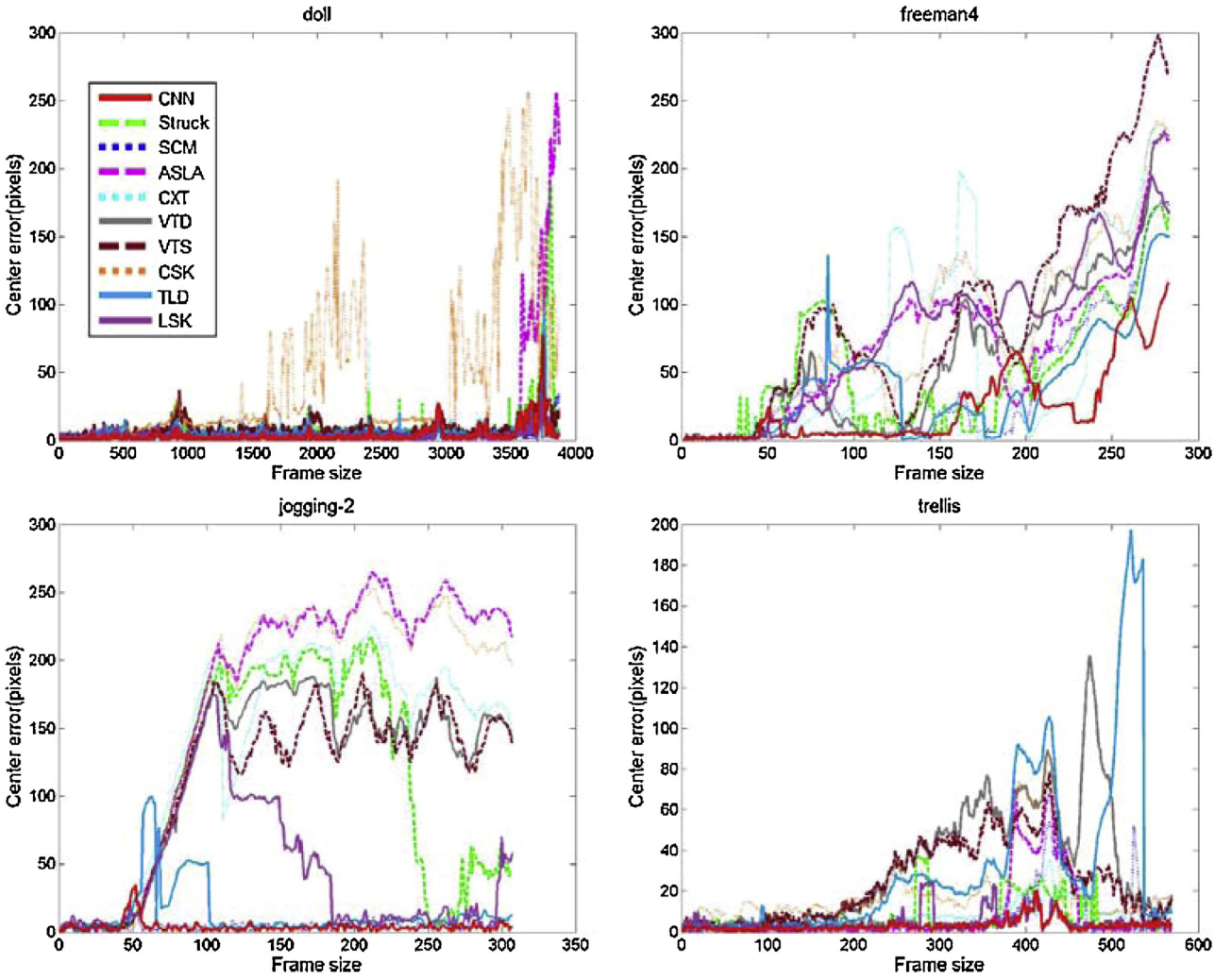

Fig. 8. Quantitative comparison on the center distance error per frame for several sequences from [51].

processor and 12 G RAM. The number of particles in particle filtering is set to 1000. The maximum buffer size T_1, the likelihood threshold T_2 and T_3 are set as 10, 0.5 and 0.8 respectively. Each image observation of the target object is normalized to a 32*32 patch. In our experiments, we choose to track only the location and size for simplicity and computational efficiency reasons. The average processing speed is about one second per frame. We use the same parameters for all of the experiments.

For performance evaluation, we test the proposed CNNTracker on the CVPR2013 tracking benchmark, which contains 50 fully annotated image sequences. Each image sequence is tagged by a number of attributes indicating to the presence of different challenging aspects, such as illumination variation, scale variation, occlusion, deformation, and background clutters, etc. In the benchmark, 30 publicly available trackers are evaluated. We follow the protocol used in the benchmark, in which the evaluation is based on two different metrics: the precision plot and success plot. The precision plot shows the percentage of frames whose estimated location is within the given threshold distance of the ground truth, and a representative precision score (threshold = 20 pixels) is used for ranking. Another metric contains the overlap precision over a range of thresholds. The overlap precision is defined as the percentage of frames where the bounding box overlap exceeds a given threshold varied from 0 to 1. In contrast to the precision plot, the trackers are ranked using the area under curve (AUC) in the success plot. Please see the original paper [51] for more details. In addition, for more close-view evaluation, we show several typical examples of the center distance error per frame with the top 10 tracker compared and the corresponding qualitative comparison results.

5.2. Comparison with other trackers

Quantitative evaluation: The quantitative comparison results of all the trackers are listed in Fig. 4 where only the top 10 trackers are shown for clarity. The values in the legend of the precision plot are the relative number of frames in the 50 sequences where the center location error is smaller than a threshold of 20 pixels. The values in the legend of the success plot are the AUC. In both the precision and success plots, the proposed CNNTracker is the state-of-the-art comparing to all alternative methods. The robustness of our CNNTracker lies in the deep CNN-based appearance model, which is discriminatively trained online to capture the high-level representation of the object and can effectively account for each variation.

Temporal and spatial robustness evaluation: it is known that a tracker may be sensitive to initialization. To analysis a tracker's robustness to initialization, we follow the evaluation protocol proposed in the CVPR2013 tracking benchmark [51]. The trackers are evaluated by perturbing the initialization temporally (referred to as temporal robustness, **TRE**) and spatially (referred to as spatial robustness, **SRE**). For TRE, each sequence is partitioned into 20 segments, whereas for SRE, 12 different bounding boxes are evaluated

for each sequence. Due to space limitation, we show only the precision and success plots for TRE in Fig. 5. The proposed CNNTracker performs favorably compared to other trackers on the temporal robustness evaluation.

Attribute-based evaluation: The object appearance variations may be caused by illumination changes, occlusions, pose changes, cluttered scenes, moving backgrounds, etc. To analyze the performance of trackers for each challenging factor, the CVPR2013 tracking benchmark [51] annotates the attributes of each sequence and constructs subsets with 11 different dominant attributes, namely: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutter and low resolution. We perform a quantitative comparison with the 30 state-of-art tracking methods on the 50 sequences annotated with respect to the aforementioned attributes. We show the precision and success plots of OPE for different subsets divided based on main variation of the target object in Fig. 6. The proposed CNNTracker achieves the state-of-the-art results for 6 out of 11 attributes: illumination variation, deformation, out-of-plane rotation, background clutter, occlusions and scale variation. For the 4 out of 11 attributes: motion blur, fast motion, in-plane rotation and out-of-view, the proposed CNNTracker performs favorably. For the low resolution attribute, the proposed CNNTracker is deteriorated due to the low quality images, which is similar to the other competing trackers.

Qualitative evaluation: Qualitative comparison with the top 10 trackers (on four typical sequences) is shown in Fig. 7. Meanwhile, for more close-view evaluation, we show the corresponding examples of the center distance error per frame in Fig. 8 with the top 10 trackers compared, which show that our method can transfer the pre-trained CNN features to the specific target object well.

According to the overall tracking results, the proposed CNNTracker achieves the state-of-the-art performance. Recall that the pre-trained CNN is learned entirely from natural scenes, which are completely unrelated to the tracking task. It implies that our method can construct robust object appearance models by effectively learning and transferring the highly general CNN features. Meanwhile, by simultaneously exploiting ground-truth and online positive samples, the drifting problem is greatly alleviated.

5.3. The impact of different training data

Since the proposed CNNTracker is pre-trained on the CIFAR-10 dataset [66], the following question arise: How the proposed method performs with different data scales? To answer the question, we investigate the performance of the proposed CNNTracker as the amount of training data grows.

Specifically, we pre-trained the proposed CNNTracker either on CIFAR-10 or tiny datatset [66]. They are denoted by CNN-10 and CNN-tiny respectively. The CIFAR-10 has 10 classes containing 6000 images each. From the 79 million tiny images, we randomly sample 202,932 images to pre-train the CNN-tiny. According the experimental results, CNN-tiny achieves a precision of 0.656 at the threshold of 20 pixels and an AUC of 0.475 on the CVPR 2013 tracking benchmark. Meanwhile, as shown in Fig. 4, CNN-10 achieves a precision of 0.654 at the threshold of 20 pixels and an AUC of 0.472 on the CVPR 2013 tracking benchmark. Obviously, the performance of the proposed CNNTracker continues to improve as data grows.

6. Conclusion

In this paper, we have proposed a robust object tracking method via deep convolutional neural network. The proposed CNNTracker does not rely on engineered features and automatically learns the most discriminative features in a data-driven way. A simple yet effective method has been used to transfer the generic and mid-level features learned from the deep CNN to the tracking task. The drifting problem is alleviated by simultaneously exploiting ground-truth and online positive samples. Moreover, a heuristic schema has been used to judge whether updating the object appearance models or not. Extensive comparison experiments on the CVPR2013 tracking benchmark demonstrate the advantage of the proposed CNNTracker.

Acknowledgments

This work is supported by Natural Science Foundation of China (No. 61202299), Natural Science Foundation of Fujian Province (No. 2015J01257), Promotion Program for Young and Middle-aged Teacher in Science and Technology Research of Huaqiao University (No. ZQN-PY210), and Outstanding Young Persons' Research Program for Higher Education of Fujian Province (No. JA13007).

References

[1] D. Comaniciu, V. Ramesh, P. Meer, Kernel-based object tracking, TPAMI (2003).
[2] M. Danelljan, F.S. Khan, M. Felsberg, J.V.D. Weijer, Adaptive color attributes for real-time visual tracking, CVPR (2014).
[3] D.A. Ross, J. Lim, R. Lin, M. Yang, Incremental learning for robust visual tracking, IJCV (2008).
[4] Q. Wang, F. Chen, W.L. Xu, M.H. Yang, Object tracking via partial least squares analysis, TIP (2012).
[5] P. Viola, M.J. Jones, Robust real-time face detection, IJCV (2004).
[6] H. Grabner, H. Bischof, On-line boosting and vision, CVPR (2006).
[7] S. Hare, A. Saffari, P. Torr Struck, Structured output tracking with kernels, ICCV (2011).
[8] R. Yao, Q.F. Shi, C.H. Shen, Y.N. Zhang, A.V.D. Hengel, Part-based visual tracking with online latent structural learning, CVPR (2013).
[9] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: application to face recognition, TPAMI (2006).
[10] V. Takala, M. Pietikainen, Multi-Object Tracking Using Color, Texture and Motion. VS, 2007.
[11] F. Yang, H. Lu, W. Zhang, G. Yang, Visual tracking via bag of features, IET Image Process. (2012).
[12] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, CVPR (2005).
[13] M. Godec, P.M. Roth, H. Bischof, Hough-based tracking of non-rigid objects, ICCV (2011).
[14] Y. Lu, T.F. Wu, S.C. Zhu, Online object tracking, learning and parsing with and-or graphs, CVPR (2014).
[15] H. Grabner, J. Matas, L.V. Gool, P. Cattin, Tracking the invisible: learning where the object might be, CVPR (2010).
[16] J.L. Fan, X.H. Shen, Y. Wu Scribble Tracker, A matting-based approach for robust tracking, TPAMI (2012).
[17] W. He, T. Yamashita, H.T. Lu, S.H. Lao, SURF tracking, ICCV (2009).
[18] F. Porikli, O. Tuzel, P. Meer, Covariance tracking using model update based on lie algebra, CVPR (2006).
[19] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, L. Bai, Real-time probabilistic covariance tracking with efficient model update, TIP (2012).
[20] X. Li, A. Dick, C.H. Shen, A.V.D. Hengel, H.Z. Wang, Incremental learning of 3D-DCT compact representations for robust visual tracking, TPAMI (2013).
[21] M. Isard, A. Blake, CONDENSATION – conditional density propagation for visual tracking, IJCV (1998).
[22] S. Avidan, Support vector tracking, TPAMI (2004).
[23] Y. Bai, M. Tang, Robust tracking via weakly supervised ranking SVM, CVPR (2012).
[24] H. Grabner, C. Leistner, H. Bischof, Semi-supervised on-line boosting for robust tracking, ECCV (2008).
[25] J. Santner, C. Leistner, A. Saffari, T. Pock, H. Bischof, PROST: parallel robust online simple tracking, CVPR (2010).
[26] J. Gall, A. Yao, L. Van, V. Lempitsky, Hough forests for object detection, tracking, and action recognition, TPAMI (2011).
[27] L. Zhang, L.V.D. Maaten, Preserving structure in model-free tracking, TPAMI (2014).
[28] X. Mei, H. Ling, Robust visual tracking using L1 minimization, ICCV (2009).
[29] C. Bao, Y. Wu, H. Ling, H. Ji, Real time robust L1 tracker using accelerated proximal gradient approach, CVPR (2012).
[30] K.H. Zhang, L. Zhang, M.H. Yang, Real-time compressive tracking, ECCV (2012).
[31] T. Zhang, B. Ghanem, S. Liu, N. Ahuja, Low-rank sparse learning for robust visual tracking, ECCV (2012).
[32] X. Jia, H.C. Lu, M.H. Yang, Visual tracking via adaptive structural local sparse appearance model, CVPR (2012).
[33] Z. Zhang, K.H. Wong, Pyramid-based visual tracking using sparsity represented mean transform, CVPR (2014).

[34] R.T. Collins, Y. Liu, M. Leordeanu, Online selection of discriminative tracking features, TPAMI (2005).
[35] B. Babenko, M. Yang, S. Belongie, Robust object tracking with online multiple instance learning, TPAMI (2011).
[36] S. Duffner, C. Garcia, Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects, ICCV (2013).
[37] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-Learning-Detection, TPAMI, 2012.
[38] B.N. Zhong, H.X. Yao, S. Chen, R.R. Ji, T.J. Chin, H.Z. Wang, Visual tracking via weakly supervised learning from multiple imperfect oracles, PR (2014).
[39] L. Fiaschi, F. Diego, K. Gregor, M. Schiegg, U. Koethe, M. Zlatic, F.A. Hamprecht, Tracking indistinguishable translucent objects over time using weakly supervised structured learning, CVPR (2014).
[40] L. Cehovin, M. Kristan, A. Leonardis, Robust visual tracking using an adaptive coupled-layer visual model, TPAMI (2013).
[41] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science (2006).
[42] G.E. Hinton, L. Deng, D. Yu, G.E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. (2012).
[43] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks, NIPS (2012).
[44] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, TPAMI (2013).
[45] R.B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, 2013, arx iv:1311.2524 [cs.CV].
[46] P. Sermanet, K. Kavukcuoglu, S. Chintala, Y. LeCun., Pedestrian detection with unsupervised multi-stage feature learning, CVPR (2013).
[47] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell Decaf, A deep convolutional activation feature for generic visual recognition, ICML (2014).
[48] A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey ACM computing surveys (2006).
[49] A.W.M. Smeulders, D.M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, M. Shah Visual, Tracking: an experimental survey, TPAMI (2014).
[50] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, A. van den Hengel, A Survey of Appearance Models in Visual Object Tracking, ACM Transactions on Intelligent Systems and Technology, 2013.
[51] Y. Wu, J.W. Lim, M.H. Yang, Online object tracking: a benchmark, CVPR (2013).
[52] A.D. Jepson, D.J. Fleet, T.F. El-Maraghi., Robust online appearance models for visual tracking, TPAMI (2003).
[53] J. Kwon, K.M. Lee, Visual tracking decomposition, CVPR (2010).
[54] S. Avidan, Ensemble Tracking, TPAMI, 2007.
[55] I. Matthews, T. Ishikawa, S. Baker, The template update problem, TPAMI (2004).
[56] T.Z. Zhang, K. Jia, C.S. Xu, Y. Ma, N. Ahuja, Partial occlusion handling for visual tracking via robust part matching, CVPR (2014).
[57] J. Kwon, K. Lee, Tracking by sampling and integrating multiple trackers, TPAMI (2014).
[58] J.M. Zhang, S.G. Ma, S. Sclaroff, MEEM robust tracking via multiple experts using entropy minimization, ECCV (2014).
[59] N.Y. Wang, D.Y. Yeung Ensemble-Based Tracking, Aggregating crowdsourced structured time series data, ICML (2014).
[60] Y. Sun, X. Wang, X. Tang, Deep Learning Face Representation by Joint Identification-Verification. Technical report, 2014, ar Xiv:1406.4773.
[61] Z. Cui, H. Chang, S.G. Shan, B.N. Zhong, X.L. Chen, Deep network cascade for image super-resolution, ECCV (2014).
[62] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006).
[63] G. Carneiro, J.C. Nascimento, Combining multiple dynamic models and deep learning architectures for tracking the left ventricle endocardium in ultrasound data, TAPMI (2013).
[64] N.Y. Wang, D.Y. Yeung, Learning a deep compact image representation for visual tracking, NIPS (2013).
[65] J.L. Fan, W. Xu, Y. Wu, Y.H. Gong, Human tracking using convolutional neural networks, TNN (2010).
[66] Krizhevsky Alex, Learning Multiple Layers of Features from Tiny Images Technique Report, 2009.
[67] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, CVPR (2015).
[68] Y. Taigman, M. Yang, M. Ranzato, L. Wolf DeepFace., Closing the gap to human-level performance in face verification, CVPR (2014).
[69] L. Deng, D. Yu, J. Platt Scalable Stacking, Learning for building deep architectures, ICASSP (2012).
[70] R. Raina, A. Madhavan, A. Ng, Large-scale deep unsupervised learning using graphics processors, ICML (2009).
[71] CUDA C Programming Guide, PG-02829-001_v5.5, NVIDIA Corporation, Santa Clara, CA, USA, 2013, July.
[72] O. Yadan, K. Adams, Y. Taigman, M. Ranzato, Multi-GPU Training of ConvNets. arX iv:1312.5853v4.
[73] V. Vanhoucke, A. Senior, M.Z. Mao, Improving the speed of neural networks on CPUs, in: Deep Learning and Unsupervised Feature Learning Workshop, NIPS, 2011.
[74] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, NIPS (2014).
[75] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, BMVC (2014).
[76] D. Laney, The Importance of 'Big Data': A Definition, Gartner, Stamford CT, USA, 2012.
[77] X.W. Chen, X. Lin, Big Data Deep Learning: Challenges and Perspectives, IEEE Access, 2014.