Real-time Deep Tracking via Corrective Domain Adaptation

Hanxi Li, Xinyu Wang, Fumin Shen, Yi Li, Fatih Porikli, Mingwen Wang

Abstract—Visual tracking is one of the fundamental problems in computer vision. Recently, some deep-learning-based tracking algorithms have been illustrating record-breaking performances. However, due to the high complexity of neural networks, most deep trackers suffer from low tracking speed, and thus are impractical in many real-world applications. Some recently-proposed deep trackers with smaller network structure achieve high efficiency while at the cost of significant decrease on precision. In this paper, we propose to transfer the deep feature which is learned originally for image classification to the visual tracking domain. The domain adaptation is achieved via some “grafted” auxiliary networks which are trained by regressing the object location in tracking frames. This adaptation improves the tracking performance significantly, both on accuracy and efficiency. The yielded deep tracker is real-time and also illustrates the state-of-the-art accuracies in the experiment involving two well-adopted benchmarks with more than 100 test videos. Furthermore, the adaptation is also naturally used for introducing the objectness concept into visual tracking. This removes a long-standing target ambiguity in visual tracking tasks and we illustrate the empirical superiority of the more well-defined task.

Index Terms—Visual tracking, deep learning, real-time

I. INTRODUCTION

Visual tracking is one of the fundamental computer vision tasks. During the last decade, as the surge of deep learning, more and more tracking algorithms benefit from deep neural networks, e.g. Convolutional Neural Networks [1], [2] and Recurrent Neural Networks [3], [4]. Despite the well-admitted success, a dilemma still existing in the community is that, deep learning increases the tracking accuracy, while at the cost of high computational complexity. As a result, most well-performing deep trackers usually suffer from low efficiency [5], [6]. Recently, some real-time deep trackers were proposed [7], [8], [9]. They achieved very fast tracking speed, but can not beat the shallow methods in some important evaluations, as we illustrate later.

In this paper, a simple yet effective domain adaptation algorithm is proposed. The equipped tracking algorithm, termed Corrective Domain Adaptation (CODA), transfers the features from the classification domain to the tracking domain, where the individual objects, rather than the image categories, are used as the learning samples. Figure 1 illustrates the main concept of the proposed domain adaptation, in a high level. The advantage of the proposed domain adaptation is three-fold. First, the shallow visual tracker, e.g., the KCF algorithm employed in this work, can extract more informative deep features from the transferred feature space. Secondly, the adaptation could be also viewed as a dimension-reduction process that removes the redundant information for tracking, and more importantly, reduces the channel number of the deep feature significantly. This leads to a remarkable increase on tracking speed. Last but not least, the adaptation introduces small auxiliary CNN “branches” that could seamlessly correct the predictions of the shallow visual trackers. Inspired by the successful adoption of objectness in visual tracking [10], [11], we exploit the category information of the tracking target in CODA, in a relatively natural way. For a certain object category, the CNN “branches” are fine-tuned to correct the tracking boxes, and thus higher tracking accuracies are obtained.

The experiments show that the proposed CODA algorithm runs in around 35 FPS while achieves comparable tracking accuracy to the state-of-the-art trackers. To our best knowledge, our CODA is the best-performing real-time visual tracker. Furthermore, given the category information of the tracking target, the corrective CNN branches lead to a significant boost in tracking accuracy while keep the tracking speed nearly unchanged. In summary, the main contribution of this paper includes:

- We propose a simple yet effective domain adaptation method for visual tracking. The adaptation not only leads to a real-time tracking speed, but also remains a high tracking accuracy which is comparable to the state-of-the-art trackers.
- For a certain type of tracking target, we propose to use the CNN branches, which are originally trained to adapt the deep feature to the visual tracking domain, to correct the initial tracking boxes. Within a sophisticated inference framework, the tracking accuracy boosts dramatically.
- From another perspective, the success of the corrective adaptation empirically proofs that a more well-defined tracking target, rather than a simple bounding-box, could benefit the tracking process significantly. In other words, this work offers an alternative to addressing the long-standing “ill-posed” problem in visual tracking.

First two authors contributed equally.

Manuscript received x.x, xxxx; revised x.x, xxxx; accepted x.x, xxxx. Date of current version x.x, xxxx. This work was supported by the National Natural Science Foundation of China (Grant No. 61672079). This paper was recommended by Associate Editor X. (Corresponding author: Fumin Shen).

H. Li and M. Wang are with the School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China (e-mail: lihanxi2001@gmail.com; mwwang@jsnu.edu.cn).

X. Wang is with Australian Institute for Machine Learning (AIML), the University of Adelaide, SA 5005, Australia. (e-mail: xinyu.wang02@adelaide.edu.au).

F. Shen is with the Center for Future Media, University of Electronic Science and Technology of China, Chengdu 611731, China. (e-mail: fumin.shen@gmail.com)

Yi. Li is with Google Brain and X, the Moonshot Factory. (e-mail: liyi.umd@gmail.com).

F. Porikli is with Research School of Engineering, Australian National University, Canberra, ACT 0200, Australia. (e-mail: fatih.porikli@anu.edu.au)
II. RELATED WORK

A. Deep trackers

Similar to other fields of computer vision, in recent years, more and more state-of-the-art visual trackers are deep-learning based. [1] is a well-known pioneering work that learns deep features for visual tracking. The DeepTrack method [10], [2] learns a deep model from scratch at the first frame and then updates it online. [12], [13] adopt the similar learning strategy, i.e., learning the deep model offline with a large number of images while updating it online for the current video sequence. [14] achieves real-time speed via replacing the slow model update with a fast inference process.

The HCF tracker [5] extracts hierarchical convolutional features from the VGG-19 network [15], then put the features into correlation filters to regress the respond map. It can be considered as a combination between deep learning and the fast shallow tracker based on correlation filters. It achieves high tracking accuracy while the speed is around 10 fps.

Hyeonseob Nam et al. proposed to pre-train deep CNNs in multi domains, with each domain corresponding to one training video sequence [6]. The authors claim that there exist some common properties that are desirable for target representations in all domains such as illumination changes. To extract these common features, the authors separate domain-independent information from domain-specific layers. The yielded tracker, termed MD-net, achieves excellent tracking performance while the tracking speed is only 1 fps.

B. Real-time deep trackers

Recently, some real-time deep trackers have also been proposed. [14] propose to infer the target location based on the deep features extracted from a fixed CNN model. Without updating the CNN model, it achieves real-time speed. In [7], David Held et al. learn a deep regressor that can predict the location of the current object based on its appearance in the last frame. The tracker obtains a much faster tracking speed (over 100 fps) comparing to conventional deep trackers. Similarly, in [8] a fully-convolutional siamese network is learned to match the object template in the current frame. It also achieves real-time speed. Even though these real-time deep trackers also illustrate high tracking accuracy, there is still a clear performance gap between them and the state-of-the-art deep trackers. [16] discusses how different regularization terms of correlation filters essentially influence the tracking performance. The yielded variations of KCF tracker achieve higher tracking accuracy than the ordinary KCF, at the cost of speed reduction (from the speed over 100 fps of the original KCF to around 37 fps).

C. Deep tracking with objectness

Nearly all the deep trackers exploit the information of generic or specific object categories to achieve higher tracking accuracies. Most of the state-of-the-art deep trackers involve the objectness implicitly via pre-training the network off-line on the dataset with object categories and bounding-boxes. [1], [14], [13], [6], [17]. [11] designs a heuristic object proposal algorithm for eliminating the non-object tracking candidates and thus the tracker can hardly lose the target due to a misleading background patch. While most of the methods mainly focus on the generic objectness, [10] pay more attention to the specific object categories. By pre-training the CNN model with the object samples from a certain category, e.g., human faces, the DeepTrack algorithm performs more robust for the specific object type.

[10] and [11] has illustrated the superiority of a more well-defined tracking task based on the introduction of objectness. In fact, the introduction partially addresses the “ill-posed”
problem in visual tracking which causes a long-standing criticism to tracking tasks. Following the high-level methodology of the pioneering work, we propose to leverage the information of tracking target category in our CODA algorithm, as we illustrate in Section IV-B.

D. Deep detectors

Similar to the visual tracking area, object detection algorithms have also been enjoying the “deep learning revolution” for a couple of years. The early successful deep detectors [18], [19] demonstrate high accuracy while the detection speed is low. Some faster deep detectors are then designed and proposed [20], [21], [22] and the most recently proposed approaches [23], [24] achieve both high detection efficiency and accuracy.

By noticing that most of the state-of-the-art visual trackers employ the “tracking-by-detection” strategy where the visual tracker is essentially a target detector. The deep detector and deep tracker are very similar from the methodology perspective. Moreover, even though there are various deep trackers and detectors, they use very limited types of neural networks, such as VGG-16/19 [15] or the residual network [25]. Those connections imply a better visual tracking algorithm can be obtained if the tracker is fused with a deep detector properly. In this paper, we enhance the deep tracking algorithm using the learning and inference strategies of a sophisticated deep detector, as we show in Section III.

III. DOMAIN ADAPTATION FOR ROBUST AND REAL-TIME VISUAL TRACKING

In this section, we introduce the details of the proposed tracking algorithm, i.e., the Corrective Domain Adaptation (CODA) tracker.

A. Network structure

The proposed work is developed based on the HCF [5] tracking algorithm which is one of the state-of-the-art visual trackers. In HCF, deep features are firstly extracted from multiple layers of the VGG-19 network [15], and a set of KCF [26] trackers are carried out on those features, respectively. The final tracking prediction is obtained in a weighted voting manner. Following the setting in [5], we also extract the deep features from conv3.5, conv4.5 and conv5.5 network layers of the VGG-19 model. However, the VGG-19 network is pre-trained using the ILSVRC dataset [27] for image classification, where the learning algorithm usually focus on the object categories. This is different from visual tracking tasks, where the individual objects are distinguished from other ones (even those from the same category) and the background. Intuitively, it is better to transfer the classification features into the visual tracking domain.

In this work, we propose to perform the domain adaptation in a simple way. A “tracking branch” is “grafted” onto each feature layer, as shown in Fig. 8. The tracking branch is actually a convolution layer which reduces the channel number by 8 times and keeps feature map size unchanged. The convolution layer is then learned via minimizing the loss function tailored for tracking, as introduced below.

B. Learn the domain adaptation

The parameters in the aforementioned tracking branch is learned following a similar manner as Single Shot MultiBox Detector (SSD), a state-of-the-art detection algorithm [22]. When training, the original layers of VGG-19 (i.e. those ones before conv5.5 are fixed and each “tracking branch” is trained independently) The flowchart of the learning procedure for one tracking branch (based on conv3.4) is illustrated in upper row of Figure 3, comparing with the learning strategy of MD-net [6] (the bottom row). To obtain a completed training circle, the adapted feature in conv5.5 is used to regress the objects’ locations and their objectness scores (shown in the dashed block). Please note that the deep learning stage in this work is purely offline and the additional part in the dashed block will be abandoned for generic object tracking. For specific categories, we propose to utilize the “tracking branches” for correcting the initial tracking boxes.

In SSD, a number of “default boxes” are generated for regressing the object rectangles. Furthermore, to accommodate the objects in different scales and shapes, the default boxes also vary in size and aspect ratios. Let \( m_{i,j} \in \{1, 0\} \) be an indicator for matching the \( i \)-th default box to the \( j \)-th ground truth box. The loss function of SSD writes:

\[
L(m, c, l, g) = \frac{1}{N} (L_{loc}(m, l, g) + \alpha L_{loc}(m, l, g))
\]

where \( c \) is the category of the default box, \( l \) is the predicted bounding-box while \( g \) is the ground-truth of the object box, if applicable. For the \( j \)-th default box and the \( i \)-th ground-truth, the location loss \( L_{loc}^{ij} \) is calculated as

\[
L_{loc}^{ij}(l, g) = \sum_{u \in \{x, y, w, h\}} m_{i,j} \cdot \text{smooth}_{L_1}(l_i^u - g_i^u)
\]

where \( g^u, u \in \{x, y, w, h\} \) is one of the geometry parameter of normalized ground-truth box.

However, the task of visual tracking differs from detection significantly. We thus tailor the loss function for the KCF algorithm, where both the object size and the KCF window size are fixed. Recall that, the KCF window plays a similar role as default boxes in SSD [26], we then only need to generate one type of default boxes and the location loss \( L_{loc}^{ij} \) is simplified as

\[
L_{loc}^{ij}(l, g) = \sum_{u \in \{x, y\}} m_{i,j} \cdot \text{smooth}_{L_1}(l_i^u - g_i^u)
\]

In other words, only the displacement \( \{x, y\} \) is taken into consideration and there is no need for ground-truth box normalization.

Note that the concept of domain adaptation in this work is different from that defined in MD-net [6], where different video sequences are treated as different domains and thus multiple fully-connected layers are learned to handle them (see Figure 3). This is mainly because in MD-net samples the training instances in a sliding-window manner. An object labeled negative in one domain could be selected as a positive sample in another domain. Given the training video number is \( C \) and the dimension of the last convolution layer is \( d_c \),
the MD-net learns $C$ independent $d_c \times 2$ fully-connected layers, i.e.,
\[ M_{fc}^i : \mathbb{R}^{d_c} \rightarrow \mathbb{R}^2, \forall i = 1, 2, \ldots, C \]  
where $M_{fc}^i, \forall i \in \{1,2,\ldots,C\}$ denotes the $C$ fully-connected layers that transferring the common visual domain to the individual object domain, as shown in Figure 3.

Differing from the MD-net, the domain in this work refers to a general visual tracking domain, or more specifically, the KCF domain. It is designed to mimic the KCF input in visual tracking (see Figure 3). In this domain, different tracking targets are treated as one category, i.e., objects. When training, the object’s location and confidence (with respect to the objectness) are regressed to minimize the smoothed $l_1$ loss. Mathematically, we learn a single mapping function $M_{conv}(\cdot)$ as
\[ M_{conv} : \mathbb{R}^{d_c} \rightarrow \mathbb{R}^4 \]  
where the $\mathbb{R}^4$ space is composed of one $\mathbb{R}^2$ space for displacement $\{x, y\}$ and one label space $\mathbb{R}^2$.

Compared with Equation 4, the training complexity in Equation 5 decreases and the corresponding convergence becomes more stable. Our experiment proves the validity of the proposed domain adaptation approach.
C. Multi-scale domain adaptation

As introduced above, the domain adaption in our CODA method is essentially a convolution layer. To design the layer, an immediate question is how to select a proper size for the filters. According to Figure 8, the feature maps from different layers vary in size significantly. It is hard to find an optimal filter size for all the feature layers. Inspired by the success of Inception network [28], we propose to simultaneously learn the adaptation filters in different scales. The response maps with different filter sizes are then concatenated accordingly, as shown in Figure 4. In this way, the input of the KCF tracker involves the deep features from different scales.

![Figure 4. Learn the adaptation layer using three different types of filters](image)

In practice, we use $3 \times 3$ and $5 \times 5$ filters for all the three feature layers. Given the original channel number is $K$, each type of filter generates $K/2$ channels and thus the channel reduction ratio is still 8 : 1.

D. Make the tracker real-time

1) Channel reduction

One important advantage of the proposed domain adaptation is the improvement of the tracking speed. It is easy to see that the speed of KCF tracker drops dramatically as the channel number increases. In this work, after the adaptation, the channel number is shrunk by 8 times which accelerates the tracker by 2 to 2.5 times.

2) Lazy feed-forward

Another effective way to increase the tracking speed is to reduce the number of feed-forwards of the VGG-19 network. In HCF, the feed-forward process is conducted for two times at each frame, one for prediction and one for model update [5]. However, we notice that the displacement of the moving object is usually small between two frames. Consequently, if we make the input window slightly larger than the KCF window, one can reuse the feature maps in the updating stage if the new KCF window (defined by the predicted location of the object) still resides inside the input window. We thus propose a lazy feed-forward strategy, which is depicted in Figure 5.

![Figure 5. The illustration of lazy feed-forward strategy. To predict the location of the object (the boy’s head), a part of the image (green window) is cropped for generating the network input. Note that the green window is slightly larger than the red block, i.e., the KCF window for predicting the current location. If the predicted location (shown in yellow) still resides inside the green lines, one can reuse the deep features by cropping the corresponding feature maps accordingly.](image)

In this work, we generate the KCF window using the same rules as HCF tracker [5], the input window is 10% larger than the KCF window, both in terms of width and height. Facilitated by the lazy feed-forward strategy, in the proposed algorithm, feed-forward is conducted only once in more than 60% video frames. This gives us another 50% speed gain.

IV. CORRECTIVE DOMAIN ADAPTATION FOR SPECIFIC OBJECT CATEGORIES

A. A long-standing ambiguity in visual tracking

Despite the widespread real-world usages, visual tracking is still criticized as less well-posed compared with other tasks with clearly-defined targets, such as object detection and semantic segmentation. In visual tracking, the only reliable target information is given at the first frame while the information could be ambiguous or misleading in many circumstances. For example, in Figure 6, a car is to be tracked in the sequence. From the viewing angle at the first frame, only the car back can be observed so it is defined as the “target” by the blue bounding box. Nonetheless, this simple target definition usually leads to an ambiguity: when the target pose changes significantly, it is hard to evaluate tracking results. In specific, as shown in Figure 6, either the yellow box or the blue box can be considered as a “perfect” tracking, depending on what exactly the tracking target is, the car back or the whole car.

![Figure 6. The commonly existing ambiguity in visual tracking. From left to right, the car back is labeled as the tracking target at the first frame, as the viewing angle changes, the car back and the visible part of the car become more and more different. Finally, when the pose changes significantly, as shown in the right column, it is hard to judge which bounding box (among blue and yellow ones) is the better tracking result.](image)

Unfortunately, a clearly-defined tracking target is usually absent in visual tracking due to the very limited information,
namely, a bounding box, given at the first frame. In this work, we try to address the ill-posed problem via imposing the object category in visual tracking tasks. In other words, the tracker tracks the object given the target’s bounding box at the first frame as well as the category of the target. This assumption is similar to the original DeepTrack algorithm [10] while we exploit the object information in a easier yet more effective way.

B. Corrective Domain Adaptation

Given the specific target category, we naturally use the proposed learning strategy proposed in Section III-B to learn a set of CNN “branches” on the samples from this category and then use the “branches” for correcting the prediction of the deep tracker. The high-level concept of the “tracking-detection-fusion” is illustrated in Figure 2

From Figure 2 one can see that the CNN model is essentially the same to that in Figure 8 expect that the auxiliary CNN branches are used for regressing the object bounding box. Note that all the regression branches are not computationally complex compared with the whole network, the extra computation burden is not heavy.

![Diagram of tracking process](image)

**Fig. 7.** The flowchart of the detection-guided tracking process. Top: the tracking box (shown in red) is obtained following the same strategy as HCF. Meanwhile, some detection bounding boxes are also generated by SSD. Bottom: after removing the unqualified detection bounding boxes, the average scale and aspect ratio of the detections are used to correct the current tracking box. Better view in color.

C. A simple yet effective guidance from detector

Given the tracking bounding-box and detection bounding-boxes, CODA merges the results in a simple yet effective way. Figure 7 demonstrates the merging process. Specifically, let us assume the tracking bounding-box (red bounding-box obtained in the same way as the ordinary HCF tracker) is represented as a 4-D vector $\mathbf{B}_t = [x_t, y_t, w_t, h_t] \in \mathbb{R}^{4 \times 1}$ where $x_t$, $y_t$, $w_t$, and $h_t$ are the $x$-axis coordinate of the box center, the $y$-axis coordinate of the box center, the width and the height of the tracking box, respectively. The SSD detector generates multiple detection bounding-boxes stored in the set $\mathcal{B}_d = \{\mathbf{B}_d^1, \mathbf{B}_d^2, \ldots, \mathbf{B}_d^N\}$ with the SSD scores $\{s_d^1, s_d^2, \ldots, s_d^N\}$. As shown in Figure 7, we firstly remove some unqualified detection boxes that are far away from the tracking box or with low scores. Normally, the qualified detection box set is selected as

$$\mathcal{B}'_d = \{\mathbf{B}_d^i \mid \text{IoU}(\mathbf{B}_d^i, \mathbf{B}_t) > 0.5 \& s_d^i > 0.6\}$$

where the function $\text{IoU}(\mathbf{B}_1, \mathbf{B}_2)$ stands for the “Intersection over Union” of two bounding boxes $\mathbf{B}_1$ and $\mathbf{B}_2$, which is used for evaluate their overlapping state.

We use $a_t = \sqrt{w_t \cdot h_t}$ and $r_t = w_t / h_t$ to represent the scale and aspect ratio of the tracking box. Suppose the number of qualified detection boxes is $N_d$, we calculate the average scale and aspect ratio for the qualified detection boxes as

$$\bar{a}_d = \frac{1}{N_d} \sum_{\mathbf{B}_d^i \in \mathcal{B}'_d} a_d^i$$
$$\bar{r}_d = \frac{1}{N_d} \sum_{\mathbf{B}_d^i \in \mathcal{B}'_d} r_d^i$$

Then the scale and aspect ratio of the final prediction, i.e., $a_t^*$ and $r_t^*$ are given by

$$a_t^* = \left( 1 - \frac{1}{1 + \exp(-\lambda(s_d^i - s_0))} \right) \cdot a_t + \frac{1}{1 + \exp(-\lambda(s_d^i - s_0))} \cdot \bar{a}_d$$
$$r_t^* = \left( 1 - \frac{1}{1 + \exp(-\lambda(s_d^i - s_0))} \right) \cdot r_t + \frac{1}{1 + \exp(-\lambda(s_d^i - s_0))} \cdot \bar{r}_d$$

where $s_d^i = \max([s_d^1, s_d^2, \ldots, s_d^N])$, i.e., the max scores over the qualified detection boxes. The hyper-parameters $\lambda$ and $s_0$ are set to 10 and 0.6 in practice.

Finally, the predicted bounding-box of CODA writes

$$\mathbf{B}_t^* = \left[ x_t, y_t, \frac{w_t \cdot a_t^*}{a_t}, \frac{w_t \cdot a_t^*}{a_t} \right].$$

From Equation 11 and Figure 7 one can see the original HCF tracking box is corrected by the detection boxes. We found the correction is usually beneficial thanks to the more clear definition of the target category and the well-learned detector. To make the corrective adaptation more clear for readers, we summarize the whole process in Algorithm 1.

V. EXPERIMENT

A. Experiment overview

In this section, we evaluate the proposed CODA tracker in two scenarios. First, the CODA for generic objects in which the corrective CNN branches are abandoned. And second, the CODA for specific target categories. The experiment is conducted on several well-adopted datasets and compared with some state-of-the-art trackers, especially the recently proposed real-time deep trackers.
The proposed CODA tracker is based on a VGG-19 network [15] which is initialized using the ILSVRC classification dataset, and then trained 3 domain adaptation layers which transfer the deep features from classification domain to tracking domain. All the experiment is implemented in MATLAB with matcaffe [29] deep learning interface, on a computer equipped with an Intel i7 4770K CPU, a NVIDIA GTX1070 graphic card and 32G RAM.

B. Experiment on generic objects

In this subsection, we report the tracking performances on generic objects of the proposed tracker and some state-of-the-art approaches. As this work focus on real-time or semi-real-time trackers, we compare our algorithm with HCF [5], GOTURN [7], the Siamese tracker [8] and the afore mentioned shallow trackers. Some well-performing shallow visual trackers, such as the KCF tracker [26], TGPR [30], Struck [31], MIL [32], TLD [33] and SCM [34] are also involved as baselines. Furthermore, two state-of-the-art deep trackers, i.e., MD-net [6] and the Siamese tracker [8] are also compared. As explained above, for generic objects, the corrective CNN branches are abandoned and only the KCF tracking results are used.

1) Results on OTB-50

Similar to its prototype [35], the Object Tracking Benchmark 50 (OTB-50) [36] consists 50 video sequences and involves 51 tracking tasks. It is one of the most popular tracking benchmarks since the year 2013. The evaluation is based on two metrics: center location error and bounding box overlap ratio. The one-pass evaluation (OPE) is employed to compare our algorithm with the HCF [5], GOTURN [7], the Siamese tracker [8] and the afore mentioned shallow trackers.

The 3 domain adaptation layers of CODA are trained on 58 video sequences that collected from VOT2013 [37], VOT2014 [38] and VOT2015 [39], excluding the ones also include in OTB50. The result curves are shown in Figure 9.

From Figure 9 we can see, the proposed CODA method beats all the competitor in the overlapping evaluation while ranks second in the location error test, with a trivial inferiority (88.01 v.s. 89.07) to its prototype, the HCF tracker. Recall that the CODA beats the HCF with a similar superiority (61.41 v.s. 60.47) and runs 3 times faster than HCF, one can consider the CODA as a major variation of the HCF, with much higher speed and maintains its accuracy. From the perspective of real-time tracking, our method performs the best in both two evaluations. To our best knowledge, the proposed CODA method is the best-performing real-time tracker in this well-accepted test.

2) Results on OTB-100

The Object Tracking Benchmark 100 is the extension of OTB-50 and contains 100 video sequences. We test our method under the same experiment protocol as OTB-50 and comparing with all the aforementioned trackers. The training set of the CODA learning keeps the same to the experiment on OTB-50. The test results as well as the tracking speeds (in fps) are reported in Table I.

As can be seen in the table, the proposed CODA algorithm keeps its superiority over all the other real-time trackers and keeps the similar accuracy to HCF. The best-performing MD-net (according to our best knowledge) enjoys a remarkable
The validity of the domain adaptation

To verify the proposed domain adaptation, here we run another variation of the HCF tracker. For each feature layer (conv4_4, conv4_4 and conv5_4) of VGG-19, one randomly selects one eighth of the channels from this layer. In this way, the input channel numbers to KCF are identical to the proposed CODA and thus the algorithm complexity of the “random HCF” and our method are nearly the same. The comparison of CODA, HCF and random HCF on OTB-50 is shown in Figure 10.

From the curves one can see a large gap between the randomized HCF (72.54 in location error and 50.68 in overlapping ratio) and the other two methods. In other words, the proposed domain adaptation not only reduce the channel number, but also extract the useful features for the tracking task.

### Algorithm 1 Corrective Domain Adaptation Tracker (CODA)

**Input:** Pre-trained CNN network $N$, video sequence $S$, init bbox $p$

1. $p_i = p$
2. $ovp = \emptyset$
3. **for each** $i \in [1, f]$ **do**
4.   $I_i = next(S)$; \Comment{Get current frame}
5.   **if** $i < I_{warm}$ **or** sum($ovp$) $> t$ **then**
6.     $feat_i, boxes_i, scores_i = forward(I_i, N)$ \Comment{Feed-forward}
7.     $boxes_i = filtering(boxes_i, scores_i, \theta)$ \Comment{Filter boxes}
8.     **else**
9.     $feat_i = forward(I_i, N')$ \Comment{Forward without fully connected layer}
10. **end if**
11. **if** $i = 1$ **then**
12.   $M_{kcf} = init(feat_i)$ \Comment{Init KCF model}
13. **else**
14.   $p_i = predict(M_{kcf}, feat_i)$ \Comment{Get predicted box}
15.   **if** boxes$_i$, then
16.     $ovp_i = overlapping(boxes_i, p_i)$
17.     **if** $ovp_i > threshold$ **then**
18.     $p_i = merging(p_i, boxes_i)$ \Comment{Predictions}
19. **end if**
20. **end if**
21. **end if**
22. **if** $i > 1$ **and** $|p_i - p_{i-1}| > \tau$ **then** \Comment{Lazy update strategy}
23.   $M_{kcf} = update(M_{kcf}, feat_i)$
24. **end if**
25. **end for each**

![Precision plots](image1)

![Location error plots](image2)

![Success plots](image3)

**TABLE I**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Ours</th>
<th>HCF</th>
<th>MD-Net</th>
<th>SiamFC</th>
<th>GOTURN</th>
<th>KCF</th>
<th>Struck</th>
<th>MIL</th>
<th>SCM</th>
<th>TLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP rate(%)</td>
<td>83.0</td>
<td>83.7</td>
<td>90.9</td>
<td>75.2</td>
<td>56.39</td>
<td>69.2</td>
<td>63.5</td>
<td>63.5</td>
<td>43.9</td>
<td>57.2</td>
</tr>
<tr>
<td>OS(AUC)</td>
<td>0.567</td>
<td>0.562</td>
<td>0.678</td>
<td>0.561</td>
<td>0.424</td>
<td>0.475</td>
<td>0.459</td>
<td>0.331</td>
<td>0.445</td>
<td>0.424</td>
</tr>
<tr>
<td>Speed(FPS)</td>
<td>34.8</td>
<td>11.0</td>
<td>1</td>
<td>58</td>
<td>165</td>
<td>243</td>
<td>9.84</td>
<td>28.0</td>
<td>0.37</td>
<td>23.3</td>
</tr>
</tbody>
</table>

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2019.2923639, IEEE Transactions on Circuits and Systems for Video Technology.
VOC is defined very differently from the concept “pedestrian” straightforwardly for pedestrians as the “person” category in Section V-B. However, the same method can not be employed the car-subset of the original training set which is introduced in Then the domain adaptation of CODA is then learned based on VOC2012 datasets [41] and keep the detection regressors. SSD-VGG-19 network for car detection on VOC2017 and applied on this category. In specific, we firstly pre-train the identical. As a result, the CODA strategy can be smoothly the VOC dataset and the involved visual tracking datasets is

C. Experiment on specific object categories

In this subsection, we test the proposed CODA tracker for specific object categories. Note that CODA can only track the object that the detector recognizes as well. The ordinary SSD is learned for predicting 20-class objects in the VOC dataset [40]. In visual tracking datasets, on the other hand, the most common categories include cars, pedestrians and human faces [36], [27]. The definition of the car category among the VOC dataset and the involved visual tracking datasets is identical. As a result, the CODA strategy can be smoothly applied on this category. In specific, we firstly pre-train the SSD-VGG-19 network for car detection on VOC2017 and VOC2012 datasets [41] and keep the detection regressors. Then the domain adaptation of CODA is then learned based on the car-subset of the original training set which is introduced in Section V-B. However, the same method can not be employed straightforwardly for pedestrians as the “person” category in VOC is defined very differently from the concept “pedestrian” in visual tracking1. We thus pre-train the SSD model on the the INRIA person dataset [42], which is a dedicated pedestrian detection dataset. Similar to the experiment on the car category, we keep the detection regressors for pedestrians and use the pedestrian-subset of the training set for domain adaptation. Finally, there is no face category in the VOC dataset and the definition of the “face” category is ambiguous among tracking tasks2 and face detection tasks [43]. Obviously, a face trackers can not be guided well by a face detector with a very different definition of “face”. Therefore, we do not conduct the experiment on the face category in this paper3. In spite of the absent face category, we still claim the experiment on cars (rigid objects) and pedestrians (non-rigid objects) is sufficient for evaluating the importance of target category information in visual tracking.

1) Results on OTB-100-Car

To evaluate the proposed method, we select all the video sequences targeting on cars from OTB-100, the totally 12 video sequences contain almost all of the tracking challenges such as scale variation, illumination variation, occlusion and motion blur. The one-pass evaluation (OPE) is employed to compare our algorithm with the HCF [5], GOTURN [7], the Siamese tracker [8] and the afore mentioned shallow trackers. The result curves are shown in Figure 11

According to Figure 11 we can see that the proposed CODA method ranks the first on location accuracy while ranks the second with the overlapping metric. It achieves significantly better results than its prototype, i.e., the HCF tracker. The performance of CODA is also very close to the best-performing MD-net. Siamese tracker is also comparable to CODA and MD-net while GOTURN performs worse than the other deep trackers. On the other hand, the shallow methods perform consistently worse than the CODA, MD-Net and the Siamese tracker.

2) Results on ILSVRC2016-VID-Car

The ILSVRC(Large Scale Visual Recognition Challenge) [27] is one of the largest visual recognition datasets. The object detection from video is a new detection task in recent years, and there are 30 basic-level categories for this task, which is a subset of the 200 basic-level categories of the object detection task. We selected 58 videos that contains car from this dataset and compare the location accuracy and overlap score over the selected deep trackers 4. As Siamese tracker and the GOTURN algorithm learned their models in this dataset, we remove them from comparison. The success plots and the precision plots are shown in Figure 12

From the figure we can see that CODA still achieves comparable accuracies as the best-performing MD-Net algorithm. The remarkable gap between the CODA’s plots and the HCF’s

Fig. 10. The location error plots and the overlapping accuracy plots of the involving trackers, tested on the OTB-50 dataset.

1 The former one includes any part of a person while the latter one usually stands for the whole human body
2 For example, in OTB-100, some videos only annotated the facial part as “face” while others involve hair and ears
3 A CODA-based face tracker can also be built based on a well-learned face detector while this is out of the scope of this paper
4 We do not involve shallow trackers in this experiment as they usually perform worse than the deep ones and the results of the shallow trackers are not directly available
Success rate

Precision

plots proofs the validity of the introduction of the object categories.

3) Results on OTB-100-Pedestrian

We perform the same experiment on the pedestrian category to evaluate the proposed corrective framework with non-rigid objects. Similar to the experimental setting shown in Section V-C1, we select all the 37 pedestrian video sequences from OTB-100 as the test set and show the tracking performances of comparing trackers in Figure 13.

According to the results shown in the Figure, one can observe a significant increase on accuracy (both for location precision and overlapping precision) after the pedestrian detector is used for correcting the tracking results. This indicates that for nonrigid objects, the proposed CODA strategy still performs well. On the other hand, the MD-Net and performs better than the CODA tracker with category information, at the cost of slow running speed (less than 1FPS).

VI. CONCLUSION AND FUTURE WORK

In this work, we propose a simple yet effective algorithm to transferring the features in the classification domain to the visual tracking domain. The yielded visual tracker, termed CODA, is real-time and achieves the comparable tracking accuracies to the state-of-the-art deep trackers. To our best knowledge, CODA is the best-performing real-time visual tracker in the literature as far.

For a specific target category, CODA guides the visual tracking by the detection results. As the deep tracker and the deep detector share most part of the deep network, no much extra computation is required. Meanwhile, we can see a dramatic performance improvement in CODA, over its prototype, the HCF tracker. This improvement implies the absence of the target category could lead to poor tracking performance while to address this ambiguity in a sophisticated way could yield much better deep trackers.

Admittedly, updating the neural network online can lift the
One possible future direction could be to simultaneously tracking accuracy significantly [2], [6]. However, the existing paper) and the shallow trackers.

Another possible direction is to involve more than one object categories in the corrective CNN branches for a certain type of tracking scenario. For instance, one can take pedestrian, car, bicycle and motorbike into consideration for each type of tracking scenario. For instance, one can take

![Precision plots](image1)

![Success plots](image2)

Fig. 13. The location error plots and the overlapping accuracy plots tested on the “pedestrian subset” of OTB-100. The comparing methods including MD-Net [6], HCF [5], the Siamese Tracker [8], GOTURN [7], CODA (this paper) and the shallower trackers.

tracking accuracy significantly [2], [6]. However, the existing online updating scheme results in dramatical speed reduction. One possible future direction could be to simultaneously update the KCF model and a certain part of the neural network (e.g. the last convolution layer). In this way, one could strike the balance between accuracy and efficiency and thus better tracker could be obtained.

Another possible direction is to involve more than one object categories in the corrective CNN branches for a certain type of tracking scenario. For instance, one can take pedestrian, car, bicycle and motorbike into consideration for road scene tracking. This could lead to even higher tracking robustness than the one-category CODA proposed in this paper.

REFERENCES


Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning


Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking


Mingwen Wang holds a Doctoral Degree (Ph.D.) in Computer Science from Shanghai Jiaotong University, China. His areas of research interest include Machine Learning, Information Retrieval, Natural Language Processing, Image Processing, and Chinese Information Processing. At present he is working as Professor, School of Computer and Information Engineering, Jiangxi Normal University, China. He is member of various professional bodies including ACL, IEEE, CCF, and ACIS.
Fig. 14. Tracking results comparison on some key frames of 9 representative OTB-100 video sequences. The comparing methods include the proposed CODA tracker (green), GOTURN [7] (blue), Siamese tracker [8] (dashed yellow), HCF tracker [5] (dashed green) and the KCF algorithm [26] (dashed light blue). The red bounding boxes are the ground-truth locations of the tracking targets. Better view in color.