Deep Location-Specific Tracking

Lingxiao Yang[1], Risheng Liu[2], David Zhang[1], Lei Zhang[1]

[1] Department of Computing, The Hong Kong Polytechnic University
[2] DUT-RU International School of Information Science & Engineering, Dalian University of Technology
{cslyang,csdzhang,cslzhang}@comp.polyu.edu.hk,rsliu@dlut.edu.cn

ABSTRACT

Convolutional Neural Network (CNN) based methods have shown significant performance gains in the problem of visual tracking in recent years. Due to many uncertain changes of objects online, such as abrupt motion, background clutter and large deformation, the visual tracking is still a challenging task. We propose a novel algorithm, namely Deep Location-Specific Tracking, which decomposes the tracking problem into a localization task and a classification task, and trains an individual network for each task. The localization network exploits the information in the current frame and provides a specific location to improve the probability of successful tracking, while the classification network finds the target among many examples generated around the target location in the previous frame, as well as the one estimated from the localization network in the current frame. CNN based trackers often have massive number of trainable parameters, and are prone to over-fitting to some particular object states, leading to less precision or tracking drift. We address this problem by learning a classification network based on 1×1 convolution and global average pooling. Extensive experimental results on popular benchmark datasets show that the proposed tracker achieves competitive results without using additional tracking videos for fine-tuning. The code is available at *https://github.com/ZjjConan/DLST*.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Tracking**;

KEYWORDS

Visual Tracking; Single Object Tracking; Location Specific Tracking; Convolutional Neural Networks

1 INTRODUCTION

Visual object tracking is a core problem in multimedia understanding and computer vision. It has numerous applications in robotics, human-computer interaction, video analysis and so on [26, 29, 38, 39]. In this paper, we focus on the problem of single object tracking. The task of single object tracking has attracted many attentions [26, 38]. A typical system is to track an arbitrary object in a video sequence, where the object has been represented in the first frame by a rectangle. Although many progress has been made in recent years, it is still a challenging task due to unknown changes of object online, such as shape deformations, occlusions, fast motion and pose variations, to name a few.

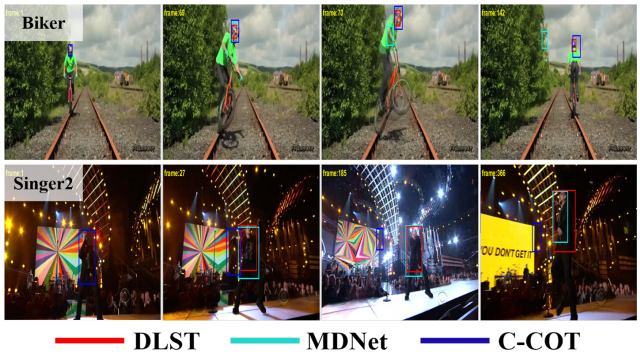


Figure 1: A comparison of the proposed DLST with MDNet [34] and C-COT [12] on two example sequences. The AUC on "Biker"/"Singer2" for DLST, MDNet and C-COT are 0.645/0.714, 0.392/0.694 and 0.531/0.081, respectively. Best viewed in color.

Many methods have been proposed for the visual tracking problem, such as multiple instance learning [2], subspace learning [32, 36, 43], ensemble learning [1, 16, 47], compressive coding [48], *etc.* In recent years, the Discriminative Correlation Filters (DCF) based methods [5, 8, 10, 19, 19] have achieved good results in terms of accuracy and speeds. Recently, the hand-crafted features utilized in these methods were replaced by more powerful deep features [14] in DCF framework [12, 30, 35] and obtained competitive tracking results. However, most of DCF based approaches tend to learn a rigid template of target object and perform poorly in the case of shape deformation (See the result of C-COT in "Singer2" video in Figure 1). More recently, several pure CNN based methods [9, 34, 44] have been developed and obtained state-of-the-art tracking results in public benchmarks [22, 23, 46].

Despite achieving promising performances, existing CNN trackers still have some drawbacks. First, to predict the position of target in the current frame, most of these trackers [20, 27, 34, 44] search the object near its location in the previous frame, which are prone to drifting in cases of fast motion and error locating in the previous frame (See the result of MDNet in "Biker" video in Figure 1). One way to mitigate this problem is to sample as many examples as possible in a larger region and feed them into a network, resulting

MM '17, October 23–27, 2017, Mountain View, CA, USA

ACM ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3123381

in very slow tracking speeds since CNN should run many times on the generated examples [34]. Second, we argue that the convention state transition strategy for visual tracking is suboptimal because they do not exploit useful information from the current frame. Third, because only limited number of positive data can be used online, many top CNN trackers [34, 44] have massive trainable parameters, and thus are likely to be over-fitting to backgrounds. To improve tracking accuracy and robustness, it is thus imperative to provide a tracker which can handle all above questions.

In this paper, we propose a simple, flexible yet effective method, *Deep Location-Specific Tracking* (DLST), which decomposes the single tracking task into localization and classification, and trains different network for each task. Our architecture is significant difference from tracker combination methods [12, 27, 30, 35, 44] because we train our networks for separated purposes: one network is for target location estimation, and the other is for foreground and background classification. Specifically, the localization network is a small network operated on a searching region which is larger than the target size, and provides a specific guess of the target location in the current frame. The classification network accepts many examples generated from the target location in the previous frame and the specific one estimated in the current frame. Additionally, our classification network is developed based on 1×1 convolution and global average pooling to reduce the problem of over-fitting. Finally, the extensive experimental results show that the proposed framework achieves competitive performance compared with other state-of-the-arts on popular benchmarks [22, 45, 46]. The remainder of this paper is organized as follow. Section 2 presents a brief review of related works, especially on the ones inspired us. Then we present our tracker in Section 3. Experimental results are shown in Section 4, followed by a conclusion of this paper in Section 5.

2 RELATED WORKS

The tracking community has been rapidly improved in recent years. A completed review of tracking methods is beyond the scope of this paper, we kindly refer interested readers to very recent surveys [26, 38]. Here, we only discuss some related works.

Multiple approaches have been developed in decades [1, 2, 16, 17, 32, 36, 43, 47, 48]. The methods of subspaces learning represent targets in a lower dimensional space and obtain good results at that time [32, 36, 43]. Struck [17] tries to minimize a structured output objective for online tracking and achieves promising results [45]. More recently, DCF based approaches have shown excellent performance in terms of accuracy and speeds on tracking benchmarks [22–24, 45, 46]. The pioneering work introduced by Bolme *et al.* [5] utilizes the DCF for adaptive tracking in grayscale image. Many extensions have been proposed to learn a robust detector for visual tracking [8, 10, 11, 19, 31]. For example, Henriques *et al.* [19] integrated DCF with HOG [7] features and achieved encouraging tracking results in popular benchmarks [24, 45]. Several existing works [10, 11] attempt to learn from part of all circular samples and obtain significant performance gain over DCF which just learns from all circular shifts. One deficiency of these trackers is that they are limited to utilize hand-crafted features, such as HOG, image color and Haar-like features, *etc.*

Since the huge success of deep CNNs in image classification [18, 25, 37, 40], many recent works have sought to represent target objects by the deep CNNs, which are pre-trained on very large scale datasets, such as ImageNet [13]. Danelljan *et al.* [9] found that with DCF tracker, the shadow layers of a pre-trained CNN performs significant better than the deeper layers as they always preserve more spatial information. In [30], Ma *et al.* trained DCF on different layers of the pre-trained CNNs, and achieved promising results in OTB [45, 46] datasets. Qi *et al.* [35] extended Ma *et al*'s work [30] and proposed an adaptive weighted method to combine different trackers. More recently, Danelljan *et al.* [12] developed a manner that employs an implicit interpolation model to pose the filter learning in continuous spatial domain. Their formulation enables efficient integration of multiple features, including different layers of CNNs, HOG, and image color, leading to state-of-the-art tracking results [22, 45, 46]. However, most DCF based trackers are prone to drift when target experiences large shape deformations and out-of-plane rotations.

The pure CNN based methods have shown strong results in tracking benchmarks [23, 45, 46]. In [27], Li *et al.* proposed multiple CNN streams to learn robust representation of the target objects. Their CNN trackers were trained from scratch online, where they suffered from lack of labeled training data. To fully exploit the representation power of CNNs, Hong *et al.* [20] utilized a pre-trained network [15] and proposed a method that combines a generative model and a discriminative classifier. The results in [20] demonstrated the effectiveness of using pre-trained CNNs. In [44], Wang *et al.* attached two specially designed networks on top of the conv5-3 and conv4-3 layers of the VGG [37] model, respectively. They proved that their crafted networks can capture complementary information for visual tracking. More recently, Nam and Han [34] proposed a multi-domain network (MDNet) and achieved top performance in visual tracking. However, their model adopted fully-connected layers as the classifier which has massive trainable parameters. To improve the performance of MDNet, they need to learn such layers from a set of labeled videos. In this paper, we propose an architecture which has fewer trainable parameters and performs favorably against state-of-the-art approaches. It can be directly employed online without using additional tracking videos for training, which largely reduces the human effort for labeling. In the next section, we will present our method.

3 OUR ALGORITHM

Our tracking framework can be decoupled into two tasks: **localization** and **classification**, as shown in Figure 2. It first crops a region at the center of target location in the previous frame, and extracts feature maps for the cropped region using a pre-trained CNN. Then a small network runs on the feature maps in a sliding-window fashion, and outputs a score map, where the max score indicates the location of target object. The bounding box of the object is obtained by back-projecting the location to the un-cropped image. Our goal in this step is to provide another specific guess in the current frame and improve the ability of tracker to accurately detect target. The location obtained by this phase is coarse because CNNs lose many spatial information due to non-identity stride used in convolution or pooling layer. The pipeline of this step is shown in Figure 2.

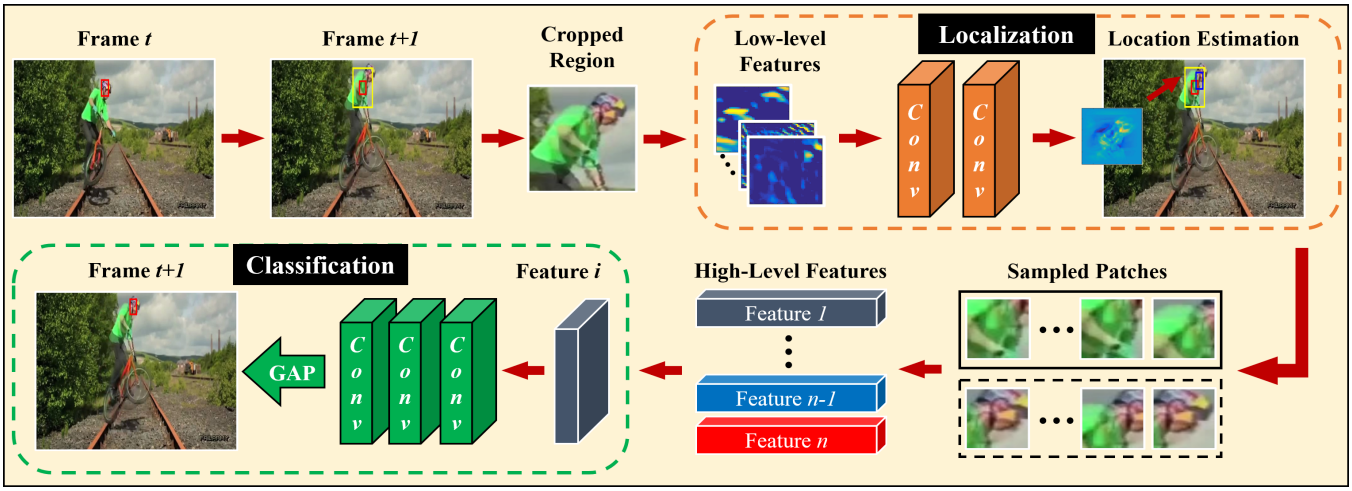


Figure 2: Overview of our architecture. The algorithm decomposes the visual tracking into localization and classification, and trains an individual network for each task. For each frame, we first crop a large region at the center of target location in the previous frame, and then utilize a small network to provide a specific location of the target in the current frame (Section 3.1). Then many candidate examples are drawn around the target location in the previous frame (patches in the black solid rectangle) and the one estimated in the current frame (patches in the black dash rectangle). The specially designed classification network accepts those examples and outputs the final target location (Section 3.2). Yellow, blue and red boxes denote the search region, the specific location and the predicted target location, respectively. Best viewed in electronic form.

Given the target location in the previous frame and the specific position from the localization network in the current frame, we sample many examples near these two positions, and feed them into another classification network to find the best one as the tracking result. The classification network is a 3 layers of convolutional neural network, followed by a global averaging pooling to get a single score for each example. This special structure is inspired by the work in [28], where the global averaging pooling is mainly utilized for regularization. We extend their idea for online tracking problem and demonstrate that the small convolutional neural network is easy to be trained with limited labeled data. In the following, we will present the details of localization, classification and our online tracking algorithm with these two components.

3.1 Localization

Our tracker is inspired from many existing methods [20, 27, 34, 44]. We begin by briefly reviewing the scheme of these trackers. Let $\boldsymbol{X}_t$ denotes many candidate states in the current frame t. $S(\boldsymbol{X}_t; \widetilde{\boldsymbol{x}}_{t-1})$ represents a state transition model which is relying on the target state in the previous frame $t-1$, where $\widetilde{\boldsymbol{x}}$ is the location predicted by the trackers or the ground truth in the initial frame. The output of S are many sampled examples of target $\boldsymbol{X}_t = \{x, y, \delta\}_{i=1}^{N}$ which are then fed into a decision model to find the best one as the target location in the frame t, where $[x, y]$, δ and N represent the center bounding box, the scale of target object and the number of sampled examples, respectively. To generate state examples $\boldsymbol{X}_t$, many studies [20, 27, 34, 44] assumed that the target states in the current frame are subject to *Normal* distribution $S_{norm}(\boldsymbol{X}_t; \widetilde{\boldsymbol{x}}_{t-1}) = \mathcal{N}(\boldsymbol{X}_t; \widetilde{\boldsymbol{x}}_{t-1}, \Sigma)$, where Σ is a diagonal covariance matrix that indicates the variance of location of sampled examples in the current frame. The normal distribution poses more weight to locations which are close to the center of target position in the previous frame. The philosophy of their methods is that objects tend to move smoothly in video sequences, and the previous location $\widetilde{\boldsymbol{x}}_{t-1}$ of object will provide a strong cue for local searching. We argue that this is a problem when: 1) the target moves too fast, and 2) the predicted location of target in the previous frame is wrong. Figure 3 shows a case of abrupt motion. As we can see, the examples generated by S_{norm} (red boxes in Figure 3 (b)) are very distant from the ground truth of the target, leading to further tracking failure.

To address above problems, we propose a localization network that provides another specific cue to improve the tracking accuracy and robustness. Firstly, we crop a large search region near the target location $\widetilde{\boldsymbol{x}}_{t-1}$ in the previous frame, and represent the cropped region by a pre-trained CNN features. Secondly, a small convolutional network (Figure 2) runs on the feature map in a sliding window fashion and outputs a score map, where the max score indicates the target location in current frame t. Finally, the estimated location, denoted as $\boldsymbol{x}_t^*$, can be obtained by back-project the target location from feature map to the original image. Note that, the localization net only adds a small computation overheads as it just forwards for a single image patch compared with exhausting searching by a classification network. As our goal is to provide a specific location of target, we set the scale of $\boldsymbol{x}_t^*$ the same to $\widetilde{\boldsymbol{x}}_{t-1}$ for simplicity. For example generation, instead of using single estimated location $\boldsymbol{x}_t^*$, we adopt S_{norm} on two centers: one is near the $\widetilde{\boldsymbol{x}}_{t-1}$, and the other is around the $\boldsymbol{x}_t^*$. The two locations are utilized for sampling as the output of localization network is not sufficient for robust tracking. This is because we cannot train an advanced network to accurately locate objects with limited labeled data online. The experiments in Section 4.2 proved that DLST performs bad when we just using

Figure 3: An illustration of two state transition methods. (a) The target in the previous frame. (b) A random subset of examples generated by a normal distribution S_{norm}. (c) A random subset of examples generated by the proposed method S_{comp}. (d) A graph shows a comparison of different S operations. The total number of examples for S_{norm} and S_{comp} is 256. For bounding boxes in (a), (b) and (c), the color green and yellow denote the ground truth and the one estimated from our localization network, while red and blue are the generated examples. Note, images shown in (a), (b) and (c) are cropped from original frames and stretched for visualization purposes (Best viewed in electronic form).

S_{norm} on estimated location. The proposed state transition strategy is denoted as $S_{comb} = \{S_{norm}(X_t; \widetilde{x}_{t-1}); S_{norm}(X_t; x_t^*)\}$. Figure 3 (c) demonstrates the effectiveness of the proposed part. It can be seen that S_{comb} can improve the probabilities of examples to cover the ground truth compared with the one using S_{norm} on the target's previous location (Figure 3 (b)). Figure 3 (d) also shows the relations between the number of sampled examples and their overlap ratio with the ground truth. In this graph, larger overlap ratio means that the examples cover more part of true target, vice verse. From this, we can conclude that the proposed S_{comb} is better than S_{norm} in this video because 1) some of examples generated by S_{comb} cover more part of true target, and 2) more number of examples are likely to cover the true target in the same overlap ratios than S_{norm}.

Details of Localization Network. The search region is 4 times larger than the target size, and it is resized to $107 \times 107 \times 3$[1] as input to the pre-trained VGG-M [6]. Here, we adopt the low-level features from the VGG-M network (first *ReLU* layer) since it preserves more spatial information. The output is a $51 \times 51 \times 96$ feature map which is then fed into our network for localization. The architecture of our network is implemented with two 1×1 convolutional layers, followed by a loss layer. The first convolutional layer has convolutional kernels of channel 96 and outputs 256 feature maps, followed by a non-linear ReLU unit and a dropout layer. The main goal of this layer is to adapt the pre-trained VGG-M features to specific videos since the same kind of objects can be treated as target or as background in different sequences. The second layer has kernels of channel 256 and outputs the heat map of the cropped region. To train the network, we employ softmax loss[2] and define labels on the feature map as:

$$y_i = \begin{cases} 2, & \text{if } ||\boldsymbol{c_t} - \widetilde{\boldsymbol{c_t}}|| \leq R, \\ 1, & \text{otherwise,} \end{cases} \tag{1}$$

where the $\boldsymbol{c_t}$ is the center of target bounding box on the feature map in the frame t. Eq (1) indicates that the samples are considered to be positive if they are within the radius R of the center of target location on the feature map. R is set by users and fixed across all videos. Also, we weight the losses by the positive and negative samples to eliminate class imbalance issue as suggested in [4].

3.2 Classification

Given a set of examples generated by S_{comb}, the next step of our algorithm is to find the target through a classification network. Generally speaking, the classification network runs on a feature map $\boldsymbol{F}$ of each example, and produces a score. A typical classification method is either using traditional decision model such as SVM [17, 20] or adopting deep classifiers like multiplayer perceptron (MLP) [34]. They both firstly vectorize the features $\boldsymbol{F}$ and then feed vectorized features into one or several fully-connected layers, followed by a SVM or softmax logistic regression layer. The recently introduced structure of MDNet [34] demonstrated that the convention MLP can be utilized as an online classifier and achieves state-of-the-art tracking results in many benchmarks.

However, since the large number of trainable parameters, the fully-connected layers are prone to over-fitting for visual tracking. Figure 4 shows an illustration of the over-fitting problem of such classifiers. The MDNet-NH has the same architecture to MDNet [34]. The results of MDNet-NH are obtained by running the published code[3], but with following 2 changes: 1) no fine-tuning on extra annotated videos and 2) using larger learning rate as it performs better than the original setting in [34] when no extra learning. From Figure 4, we can see that MDNet-NH tends to track some particular regions, such as the right corner of the screen in the "Singer2" video, and the "gitar" in the "Shaking" sequence. The detailed quantitative comparisons are shown in Section 4.

A possible way to address this problem is to train these layers on many annotated videos such as OTB [45, 46] and VOT [22–24] (See MDNet results in green bounding boxes in Figure 4). Our goal in this paper is to alleviate the problem of over-fitting without using any extra videos, which largely reduces the human effort for labeling. Our specially designed tracker consists of three 1×1 convolutional layers and outputs a score map of the same size to $\boldsymbol{F}$. The global average pooling (GAP) [28] is adopted and its output is

[1] RGB channels

[2] For simplicity we implement it as a two-class softmax layer. Alternatively, one may use any other classifier, such as logistic regression to produce score map.

[3] https://github.com/HyeonseobNam/MDNet

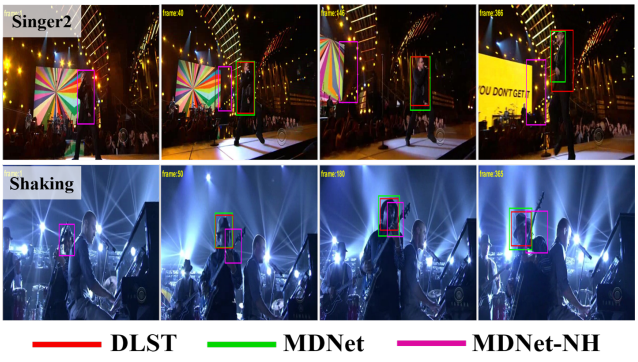


Figure 4: An illustration of the proposal method compared with other two baselines. The MDNet-NH adopts the same architecture to MDNet [34] for visual tracking. However, MDNet-NH do not fine-tuning the pre-trained ImageNet model [6] by using additional tracking videos. In addition, the compared DLST do not utilize the localization phase to assist tracking.

then directly fed into a softmax layer to produce final prediction for each example. Figure 4 demonstrates that our tracker, with less trainable parameters than the ones used in MDNet and MDNet-NH models (0.35M *v.s.* 2.6M), can successfully track the target, even without any extra pre-training. Note that, in this comparison, we do not utilize the proposed localization stage. Therefore, these gains are solely because of our specially designed architecture compared with the structure of MDNet-NH. More comparison details can be found in Section 4.

Details of Classification Network. The sampled examples are resized to $107\times107\times3$ pixels. In this paper, the $relu_3$ layer of the pre-trained VGG-M [6] model is utilized as feature extractor because it captures more semantic part or category information than lower layers. For each example, the output $3\times3\times512$ dimensional feature map is then fed into our classification network for final prediction. Our specially designed classification network consists of three 1×1 convolutional layers. The first two convolutional layers have 512 and 128 number of filters, respectively. The last one produces 2 outputs for target and background. For network learning, we again employs softmax logistic regression for simplicity. The target state in the current frame is given by finding the example with the maximum positive score as $\widetilde{\boldsymbol{x}}_t = arg\,max_{x_i\in\boldsymbol{X}_t} f^+(x_i)$ [34], where f^+ is the output positive score from our classification network.

3.3 Online Tracking and Update

Online Tracking. The pipeline of our algorithm is detailed in Figure 2. For each frame, we first adopt the localization network to provide another specific location and use S_{comb} to generate examples. Then the example with max output from the classification network is chosen as target state $\widetilde{\boldsymbol{x}}$ in the current frame t. The practical considerations of using these two networks for tracking can be found in Section 3.1 and Section 3.2, respectively.

Model Updates. We utilize the same strategy to update our two networks. In detail, we adopt short-term and long-term strategies for tracker robustness and adaptiveness, respectively [34]. Long-term updates are conducted using the positive examples collected for $\mathcal{T}_{long}$ regular period of time, while short-term updates are executed when the output of our classification network is below some threshold (0.5 used in this paper). The positive examples selected for short-term learning comes from a $\mathcal{T}_{short}$ period of time. All the negative examples are selected from the $\mathcal{T}_{short}$ period of time. Note that, we freeze the feature extractor networks both in localization and classification phases for tracking.

Bounding Box Regression & **Hard Negative Mining.** These two strategies are often used in object detection problem [15]. Recently, Nam and Han [34] demonstrated that bounding box regression and hard negative mining can largely improve the tracking precision and robustness. In this paper, we also adopt the same manners for our tracker. Specifically, the box regressor is just trained using examples sampled in the first frame and applied in the subsequent each frames if the predicted targets are beyond given threshold ($f^+ > 0.5$). For classification network, the hard negative examples are selected based on the top positive scores to improve the discriminativeness. Note that, we do not use these strategies for localization network.

4 EXPERIMENTS

We validate the proposed DLST on 3 popular tracking datasets: OTB50 [45], OTB100 [46] and VOT2016 [22]. For algorithm analysis, we quantitatively evaluate trackers using overlap success (OS) and distance precision (DP) rate on OTB100 dataset. On OTB dataset, the OS is defined as the percentage of frames in a video where the IoU overlap ratio is larger than given threshold (0.5). DP is computed by averaging Euclidean distance between the center locations of the target objects and the annotated ground truths for a video. We use the standard threshold 20 pixels in DP. We also present the area-under-the-curve (AUC), where the mean OS over all videos is drawn over the range of [0, 1]. We follow the testing protocol in [46] and use the same parameters for all sequences. The one-pass evaluation (OPE) is employed in this paper. For VOT2016 testing, we use the official toolkit[4] and report results using expected average overlap (EAO), robustness (failure rate, FRT) and accuracy (ACC). We kindly refer authors to [22, 46] for more detailed description about these datasets, evaluation protocol, and evaluation metrics.

4.1 Implementations

For localization network, we apply a cosine window [5] on the input features, extracted by the pre-trained VGG-M [6], to remove the boundary discontinuities of learning filters. The radius R in Eq (1) is 8. The response map output from localization network is upsampled 2 times for more precise localization, as suggested in [4]. The network learning is achieved using stochastic gradient descent with momentum technique. In details, the learning rate is 0.03 for the last convolutional layers in both localization and classification network. For other layers in these two networks, we use 0.003 to train the models. The maximal iteration is 30 for training both networks in the initial frame, and 10 for online update. The momentum and weight decay are set to 0.9 and 0.0005, respectively.

[4]https://github.com/votchallenge/vot-toolkit

Table 1: Comparisons with baseline trackers on the entire 100 benchmark sequences. We report distance precision (DP) rate at a threshold of 20 pixels, and overlap success (OS) rate at an overlap threshold of 0.5 for all compared methods under different attributes annotated by the authors [46]. The attributes are illumination variation (IV), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-view (OV), background cluttered (BC) and low resolution (LR). The trackers without localization network, with fully-connected layers, and just using estimated location for tracking are denoted as "nloc", "fc" and "sloc", respectively. The first and second best values are highlighted by bold and underline.

	Method	Mean	IV	OPR	SV	OCC	DEF	MB	FM	IPR	OV	BC	LR
DP rate (%)	DLST	**91.4**	**91.9**	**91.0**	**90.1**	86.6	**88.9**	**89.6**	**89.0**	**91.4**	**87.7**	**92.3**	**94.9**
	DLST-sloc	75.8	78.2	75.3	74.3	68.3	72.1	73.1	69.8	74.0	65.9	75.3	60.1
	DLST-fc	89.5	86.4	88.7	88.6	**87.1**	87.0	87.7	87.3	87.2	84.4	85.5	87.8
	DLST-nloc	88.8	89.4	88.2	86.7	82.5	86.4	85.3	84.6	89.8	79.3	88.6	81.6
	MDNet-NH	88.2	87.0	86.6	87.1	86.4	84.9	87.2	87.4	87.6	84.6	86.7	89.9
	MDNet-NL	78.7	74.3	78.7	75.9	72.9	78.3	74.3	75.7	76.4	65.4	68.8	72.5
OS rate (%)	DLST	**85.0**	**86.6**	**83.6**	**80.7**	81.0	**80.1**	85.8	**82.1**	**82.4**	**78.7**	**85.1**	**79.6**
	DLST-sloc	69.1	73.7	66.7	66.0	61.4	63.7	70.6	63.8	64.7	61.6	69.1	52.2
	DLST-fc	83.5	82.1	80.5	79.4	**81.6**	79.2	85.0	81.8	78.9	77.7	77.4	74.6
	DLST-nloc	83.3	85.3	81.8	78.4	77.7	78.6	83.6	79.8	81.2	72.4	83.2	69.8
	MDNet-NH	81.7	82.4	79.9	78.8	80.9	77.6	**85.9**	81.8	79.3	76.9	79.5	73.8
	MDNet-NL	73.3	70.6	71.8	68.5	68.3	70.6	74.0	72.0	68.8	58.7	65.2	60.7

For classification network training, we construct a mini-batch with 128 examples, consisting of 32 positives and 96 negatives, where the negatives are selected from a larger batch (1024). While, for localization network learning, the mini-batch size is 8.

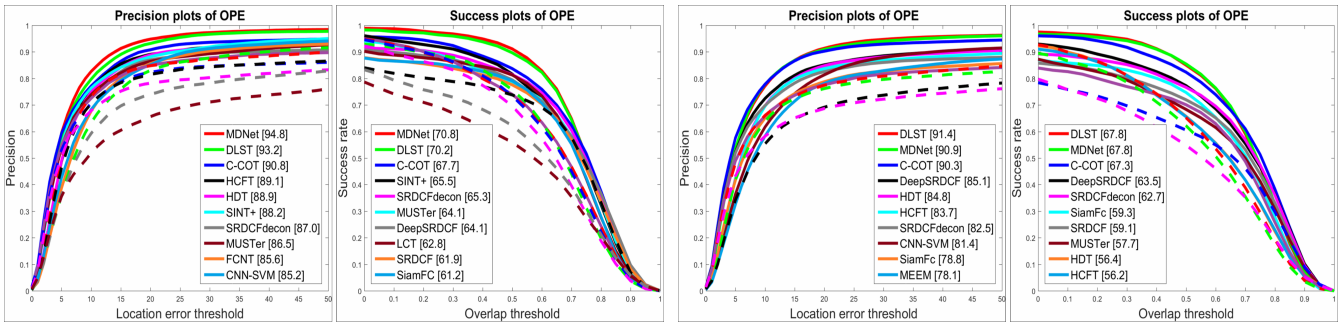
To generate target candidates $\boldsymbol{X}_t$ in each frame, we sample 128 examples in translation and scale dimension around the previous frame, and near the estimated one respectively. Therefore, the total number of output of $\boldsymbol{S}_{comb}$ is 256. The mean of normal distribution in $\boldsymbol{S}_{comb}$ is the target location in the last frame, and the other one estimated in the current frame. In fact, we set the estimated $\boldsymbol{x}_t^*$ the same to the target state in the previous frame $\widetilde{\boldsymbol{x}}_{t-1}$ if the maximum score output from localization network is less then 0.5. In this case, the proposed $\boldsymbol{S}_{comb}$ is degraded into convention state transition $\boldsymbol{S}_{norm}$. The Σ is a diag matrix $diag(0.09r^2, 0.09r^2, 0.25)$, where r is the mean of target size in the last frame. The scale of each candidate is computed by the initial target scale mutiplying 1.05^{δ_i}.

For training data generation, we collect 50 (500) positive and 200 (5000) negative samples from every successful tracked frames (or the initial frame). The positive and negative examples have $\geq$ 0.7 and $\leq$ 0.3 ($\leq$ 0.5) IoU overlap ratios with the target state $\widetilde{\boldsymbol{x}}_t$ (or $\widetilde{\boldsymbol{x}}_1$), respectively. For bounding-box regression, we use 1000 training examples. In order to learn a robust localization network, we randomly select 5 (50) positive samples from the generated examples, and crop searching regions centered with these positives (or the ground truth in the first frame). We store $relu3$ and $relu1$ features for online learning since feature extractors are frozen. The time periods for long-term $\mathcal{T}_{long}$ and short-term $\mathcal{T}_{short}$ are set to 100 and 20 respectively. Note that all settings are similar to the ones utilized in [34], except for the state transition. With above settings, our tracker is implemented in Matlab using MatConvNet [42] toolbox and runs at around 1 frames per second on an Intel I7-4790k 4.0 GHz CPU and a NVIDIA GTX 980 GPU.

4.2 In-depth Studies

In this section, we analyze the proposed tracker DLST on OTB100 benchmark by demonstrating the effectiveness of different integrated components, including the method 1) without using localization network (DLST-nloc), 2) without our specially designed classifier (DLST-fc), and 3) just using estimated location for tracking (DLST-sloc). As our work is a pure CNN based method, similar to recently developed tracker – MDNet [34], we also include two variants of it. For the sake of fair comparisons, all methods are not fine-tuned using additional labeled tracking videos. In Table 1, **N** stands for no additional training for MDNet, while **L** and **H** mean the method using original learning rate in the paper [34] and the approach with higher learning rate (×10). The results of MDNet-NL and MDNet-NH are obtained by the published code from the authors. From Table 1, we can conclude that our full model **DLST** attains the highest score in terms of mean DP and mean OS among all compared methods.

As presented in Table 1, because of no extra fine-tuning, the MDNet-NH with higher learning rate is significant better than MDNet-NL in these two evaluation metrics. The DLST-nloc achieves similar results with MDNet-NH in DP, but performs better in OS. Note that, the DLST-nloc do not use localization network to provide another specific location. Therefore, the performance gain is solely because of our specially designed classifier. For more details, the DLST-nloc achieves better performance over MDNet-NH in cases of IV, OPR, DEF, IPR and BC. This is because our tracker has fewer trainable parameters and is less prone to over-fitting to particular regions of target objects, such as the "gitar" in Figure 4. Another reason is that the GAP encourages the network invariant of spatial translations of the input by averaging activations from all spatials. This might help the classifier to identify partial of target objects. For example, the human in "Singer2" of Figure 4 experiences large



(a) Results for OTB50 sequences [45] **(b) Results for OTB100 sequences [46]**

Figure 5: Quantitative results on the benchmark datasets [45, 46]. The values in the legend represent the precisions at a threshold of 20 pixels for precision plots, and the area-under-curve (AUC) scores for success plots. Only the top 10 trackers are shown in the legend for clarity. Best viewed in electronic form.

deformations and some part of his body is missing. In this video, the DLST-nloc achieves 95.4/86.9 (OS/DP). However, MDNet-NH fails to track the objects, only achieving 3.6 OS and 3.6 DP. In addition, we conduct an experiment to prove the effectiveness of our localization phase. The tracker is denoted as DLST-fc and has the same structure of classifier to MDNet-NH. In this setting, we adopt our localization network for tracking. The results from Table 1 shows that DLST-fc attains better performance in OS and DP, especially in the case of OCC. This demonstrates that our localization network can provide specific conjectures for targets in the current frame and thus can help our tracker to successful identify the partial of targets or recover from error locations. We also conducted another study on DLST when only the estimated location is utilized for tracking, which is denoted as DLST-sloc in Table 1. As shown from this table, the output of localization is not sufficient for successful tracking. This proves the strength of the proposed state transition strategy S_{comb}, which retains two centrals for tracking. By using both proposed componenets, our tracker can track target objects under a variety of conditions, achieving maximum performance. According to above analysis, we use our full model DLST, including localization and our specially designed classification network in the rest of comparisons.

4.3 Comparisons with State-of-the-arts

OTB 50&100 Datasets. We compare our DLST with many state-of-the-art methods: Struck [17], KCF [19], DSST [8], LCT [31], MEEM [47], MUSTer [21], CNN-SVM [20], FCNT [44], SINT+ [41], HCF [30], HDT [35], SRDCF [10], SRDCFadcon [11], deepSRDCF [9], Staple [3], SiameseFc [4], C-COT [12] and the original MDNet [34]. Many of aforementioned trackers are developed based on deep neural networks, such as HCF [30], C-COT [12], deepSRDCF [10] and MDNet [34], *etc.*

Figure 5 illustrates the precision and AUC on center location error and bounding box overlap ratio, respectively. Our tracker – DLST – performs favorably against all compared methods. Specifically, we achieve 67.8 and 91.4 scores of AUC and precision for success and precision plots on OTB100 dataset, respectively. While on OTB50 benchmark, our tracker attains 70.2 AUC scores and 93.2 precisions. The results from these two benchmarks are similar to the ones obtained by MDNet. However, our DLST can be employed directly for visual tracking on any videos without additional training as MDNet. In comparisons to the methods using DCF-CNN framework, our DLST still perform better than the second best one C-COT in terms of AUC and precisions on these two benchmarks. Moreover, the C-COT utilizes diverse features for tracking, such as color name, HOG, and activations from multiple CNN layers. Instead, we just use two CNN layers for different purposes. Figure 5 also shows that the existing trackers heavily rely on deep CNNs, except for SRDCF, SRDCFdecon, MUSTer and LCT. The results clearly demonstrate that the deep CNNs play important roles in the visual tracking task while obtaining high performance gain.

Table 2: State-of-the-art results on VOT2016 dataset. Only the top trackers are shown. EAO: expected average overlap; ACC: accuracy; FRT: robustness (failure rate).

	EBT [49]	DDC [22]	Staple [3]	MLDF [22]	SSAT [22]	TCNN [33]	C-COT [12]	DLST ours
EAO	0.291	0.293	0.295	0.311	0.321	0.325	0.331	**0.343**
ACC	0.44	0.53	0.54	0.48	**0.57**	0.54	0.52	0.55
FRT	0.90	1.23	1.35	**0.83**	1.04	0.96	0.85	**0.83**

Figure 6 shows the results under different video attributes annotated in the benchmark [45]. As shown in Figure 6, our tracker effectively handles many kinds of challenging situations. In particular, the AUC and precisions plots scores of our tracker in the case of low-resolution are significant better than all other compared methods, such as SiamFc [4], MDNet [34] and C-COT [12]. Interestingly, from Table 1, we can see that none of each component of our algorithm performs well in low-resolution videos. For example, DLST-nloc and DLST-fc perform not as well as MDNet-NH. But the method that combines each of our component attains largely improvement over other compared approaches (as shown in Table 1). For other challenging attributes, our DLST still perform better among all compared state-of-the-art methods. These results demonstrate that the importance of the combinations. In addition to successful results, we show two cases of failure tracking in Figure 7.

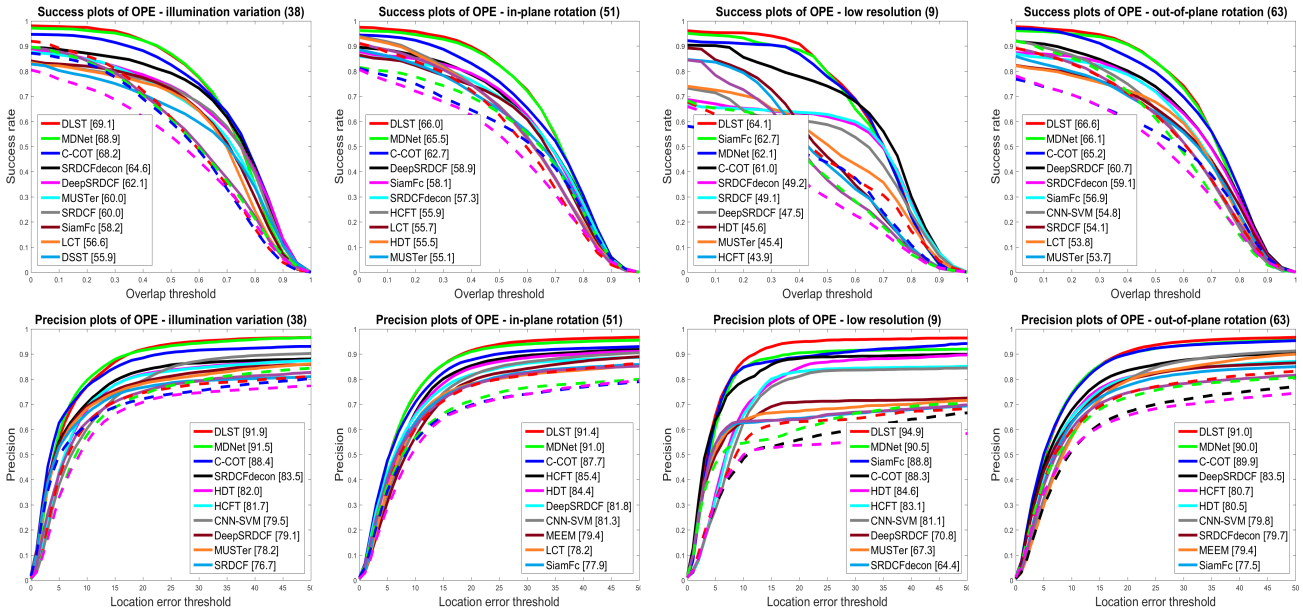


Figure 6: The precisions and AUC for success plots for four challenge attributes: illumination variation, in-plain rotation, low resolution and out of view, and scale variation. Only the top 10 trackers are shown in the legend for each attributes.

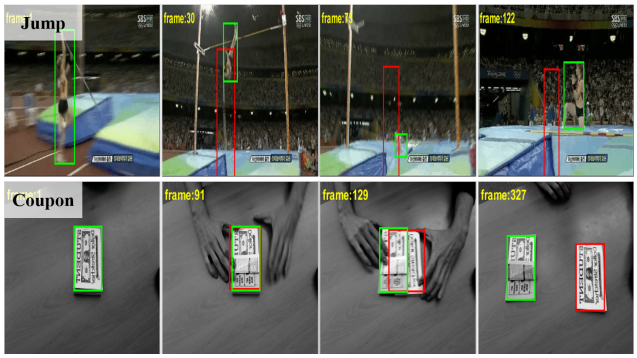


Figure 7: Failure cases of our method on "Jump" and "Coupon". Green and red bounding mean the ground truth and the results from our tracker, respectively.

Specifically, our DLST fails to track target objects when they have very similar appearance (*e.g.* result in "Coupon" sequence) and experience dramatic topology changes (*e.g.* result in "Jump" sequence). Another limitation of our tracker is that the running speeds (1 fps on OTB datasets) are far below real-time usage, which cannot be easily employed in other products, such as mobile phone and many embedded devices. We leave these issues for further studies.

VOT2016 Dataset. We also conducted an experiment on VOT 2016 [22] challenge dataset. In this dataset, we adopt the same network structure and the same hyper-parameters as we do on OTB datasets. The results are shown in Table 2. As compared with other top-ranked trackers, our tracker DLST achieves the best performance with an EAO score of 0.343. Specifically, both MLDF [22] and our tracker DLST obtain the first place in the failure rate (FRT) among all compared performers. For accuracy evaluation, the best result is obtained by SSAT [22], which extends the framework of MDNet [34] by incorporating two advanced techniques: segmentation and occlusion prediction. Without such techniques, our tracker DLST still attain the second score (0.55) among all top-ranked trackers. We believe the techniques utilized in SSAT can also be employed in our framework to improve the tracking robustness and accuracy.

5 CONCLUSIONS

In this paper, we presented a novel tracking algorithm, namely Deep Location-Specific Tracking (DLST), which decouples the single tracking task into a localization task and a classification task, and trains a separated network for each task. The localization network exploits the information from the current frame, and provides a specific location for state transition, which is helpful to improve the performance of tracking. Our classification network has fewer trainable parameters due to its 1×1 convolutional layers and the global average pooling. This specially designed structure is less likely to over-fitting, and improves the tracking accuracy and robustness. The proposed DSLT algorithm achieves competitive results on the benchmarks without using additional annotated videos, largely reducing the human effort for labeling.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the HK RGC General Research Fund (PolyU 152240/15E) and National Natural Science Foundation of China (grant no. 61672446 and 61672125).

REFERENCES

[1] Shai Avidan. 2007. Ensemble tracking. *IEEE TPAMI* 29, 2 (2007).

[2] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2009. Visual tracking with online multiple instance learning. In *CVPR*. IEEE, 983–990.

[3] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. 2016. Staple: Complementary learners for real-time tracking. In *CVPR*. IEEE, 1401–1409.

[4] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. 2016. Fully-Convolutional Siamese Networks for Object Tracking. *arXiv preprint arXiv:1606.09549* (2016).

[5] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. 2010. Visual object tracking using adaptive correlation filters. In *CVPR*. IEEE, 2544–2550.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. 2014. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *BMVC*. BMVA Press. arXiv:cs/1405.3531

[7] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *CVPR*, Vol. 1. IEEE, 886–893.

[8] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg. 2014. Accurate scale estimation for robust visual tracking. In *BMVC*. BMVA Press.

[9] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. 2015. Convolutional features for correlation filter based visual tracking. In *ICCVW*. IEEE, 58–66.

[10] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. 2015. Learning spatially regularized correlation filters for visual tracking. In *ICCV*. IEEE, 4310–4318.

[11] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. 2016. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*. IEEE, 1430–1438.

[12] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*. Springer, 472–488.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*. IEEE, 248–255.

[14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition.. In *ICML*. 647–655.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*. IEEE, 580–587.

[16] Helmut Grabner, Michael Grabner, and Horst Bischof. 2006. Real-time tracking via on-line boosting.. In *BMVC*, Vol. 1. BMVA Press, 6.

[17] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. 2016. Struck: Structured output tracking with kernels. *IEEE TPAMI* 38, 10 (2016), 2096–2109.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. IEEE, 770–778.

[19] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. 2015. High-speed tracking with kernelized correlation filters. *IEEE TPAMI* 37, 3 (2015), 583–596.

[20] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. 2015. Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network.. In *ICML*. 597–606.

[21] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. 2015. Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking. In *CVPR*. IEEE, 749–758.

[22] Matej Kristan, Aleš Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin, Tomas Vojir, Gustav Häger, Alan Lukežič, and Gustavo Fernandez *et al.* 2016. The Visual Object Tracking VOT2016 challenge results. Springer. (Oct 2016). http://www.springer.com/gp/book/9783319488806

[23] Matej Kristan, Jiri Matas, Aleš Leonardis, Michael Felsberg, Luka Čehovin, Gustavo Fernandez, Tomas Vojir, Gustav Häger, Georg Nebehay, and Roman Pflugfelder *et al.* 2015. The Visual Object Tracking VOT2015 challenge results. In *Visual Object Tracking Workshop 2015 at ICCV2015*. IEEE.

[24] Matej Kristan, Roman Pflugfelder, Aleš Leonardis, Jiri Matas, Luka Čehovin, Georg Nebehay, Tomas Vojir, Gustavo Fernandez, Alan Lukežič, and Aleksandar Dimitriev *et al.* 2014. The Visual Object Tracking VOT2014 challenge results. (2014). http://www.votchallenge.net/vot2014/program.html

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.

[26] A Li, M Lin, Y Wu, MH Yang, and S Yan. 2016. NUS-PRO: A New Visual Tracking Challenge. *IEEE TPAMI* 38, 2 (2016), 335–349.

[27] Hanxi Li, Yi Li, and Fatih Porikli. 2014. Robust online visual tracking with a single convolutional neural network. In *ACCV*. Springer, 194–209.

[28] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[29] Xiaobai Liu. 2016. V3I-STAL: Visual Vehicle-to-Vehicle Interaction via Simultaneous Tracking and Localization. In *MM*. ACM, New York, NY, USA, 1117–1126. https://doi.org/10.1145/2964284.2964285

[30] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. 2015. Hierarchical convolutional features for visual tracking. In *ICCV*. IEEE, 3074–3082.

[31] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. 2015. Long-term correlation tracking. In *CVPR*. IEEE, 5388–5396.

[32] Xue Mei and Haibin Ling. 2009. Robust visual tracking using $l1$ minimization. In *ICCV*. IEEE, 1436–1443.

[33] Hyeonseob Nam, Mooyeol Baek, and Bohyung Han. 2016. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242* (2016).

[34] Hyeonseob Nam and Bohyung Han. 2016. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*. IEEE, 4293–4302.

[35] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang. 2016. Hedged deep tracking. In *CVPR*. IEEE, 4303–4311.

[36] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. 2008. Incremental learning for robust visual tracking. *IJCV* 77, 1 (2008), 125–141.

[37] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

[38] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. 2014. Visual tracking: An experimental survey. *IEEE TPAMI* 36, 7 (2014), 1442–1468.

[39] Michael Stengel, Steve Grogorick, Martin Eisemann, Elmar Eisemann, and Marcus Magnor. 2015. An Affordable Solution for Binocular Eye Tracking and Calibration in Head-mounted Displays. In *MM*. ACM, 15–24.

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR*. IEEE, 1–9.

[41] Ran Tao, Efstratios Gavves, and Arnold W M Smeulders. 2016. Siamese Instance Search for Tracking. In *CVPR*. IEEE.

[42] Andrea Vedaldi and Karel Lenc. 2015. Matconvnet: Convolutional neural networks for matlab. In *MM*. ACM, 689–692.

[43] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. 2013. Online object tracking with sparse prototypes. *IEEE TIP* 22, 1 (2013), 314–325.

[44] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. 2015. Visual tracking with fully convolutional networks. In *ICCV*. IEEE, 3119–3127.

[45] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. 2013. Online object tracking: A benchmark. In *CVPR*. IEEE, 2411–2418.

[46] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. 2015. Object tracking benchmark. *IEEE TPAMI* 37, 9 (2015), 1834–1848.

[47] Jianming Zhang, Shugao Ma, and Stan Sclaroff. 2014. MEEM: robust tracking via multiple experts using entropy minimization. In *ECCV*. Springer, 188–203.

[48] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. 2012. Real-time compressive tracking. In *ECCV*. Springer, 864–877.

[49] Gao Zhu, Fatih Porikli, and Hongdong Li. 2016. Beyond local search: Tracking objects everywhere with instance-specific proposals. In *CVPR*. IEEE, 943–951.