Learning Dynamic Siamese Network for Visual Object Tracking

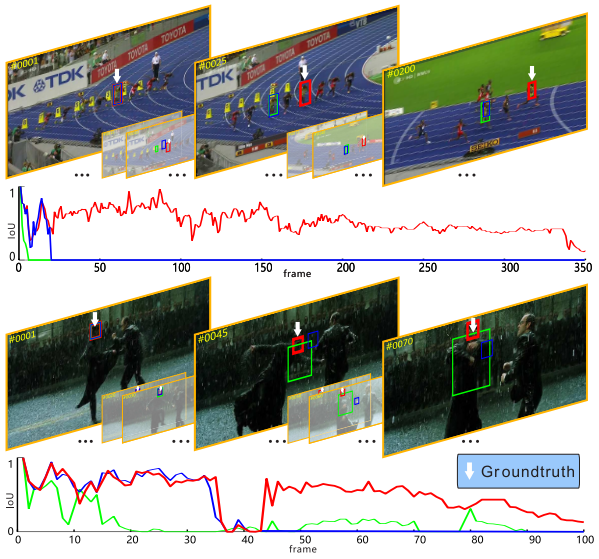
Qing Guo[1,3], Wei Feng[1,3]*, Ce Zhou[1,3], Rui Huang[1,3,5], Liang Wan[2,3], Song Wang[1,3,4]
[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
[2] School of Computer Software, Tianjin University, Tianjin, China
[3] Key Research Center for Surface Monitoring and Analysis of Cultural Relics, SACH, China
[4] University of South Carolina, Columbia, SC 29208, USA
[5] School of Computer Science and Technology, Civil Aviation University of China
{tsingqguo,wfeng,zhouce,ruihuang,lwan}@tju.edu.cn, songwang@cec.sc.edu

Abstract

How to effectively learn temporal variation of target appearance, to exclude the interference of cluttered background, while maintaining real-time response, is an essential problem of visual object tracking. Recently, Siamese networks have shown great potentials of matching based trackers in achieving balanced accuracy and beyond real-time speed. However, they still have a big gap to classification & updating based trackers in tolerating the temporal changes of objects and imaging conditions. In this paper, we propose dynamic Siamese network, via a fast transformation learning model that enables effective online learning of target appearance variation *and* background suppression *from previous frames. We then present* elementwise multi-layer fusion *to adaptively integrate the network outputs using multi-level deep features. Unlike state-of-the-art trackers, our approach allows the usage of any feasible generally- or particularly-trained features, such as SiamFC and VGG. More importantly, the proposed dynamic Siamese network can be jointly trained as a whole directly on the labeled video sequences, thus can take full advantage of the rich spatial temporal information of moving objects. As a result, our approach achieves state-of-the-art performance on OTB-2013 and VOT-2015 benchmarks, while exhibits superiorly balanced accuracy and real-time response over state-of-the-art competitors.*

1. Introduction

Visual tracking aims to track an arbitrary temporally-changing object, with the target being only specified at the first frame. Since potential changes of the object and its context are basically unknown and constantly happen, this problem, being highly useful for many computer vision tasks, such as surveillance, video analysis and augmented



Trackers	AUC in OTB 2013	Speed (fps)
DSiam (ours)	0.642	45
DSiamM (ours)	0.656	25
SiamFC [2]	0.608	86
GOTURN [15]	0.447	100
MDNet [22]	0.708	1
R-MDNet	0.619	1
DeepSRDCF [6]	0.641	1

Figure 1. Two tracking examples of state-of-the-art matching based trackers (GOTURN [15] and SiamFC [2]), deep classification & updating based trackers (MDNet [22] and DeepSRDCF [6]), and the proposed approach (DSiam). The intersection over union (IoU) between ground truth and tracking result of each tracker on each frame is shown. Previous two matching based methods easily miss the object when similar objects coexist (the 1st case) or target changes significantly (the 2nd case). The table shows the average tracking speed, measured on NVIDIA TITAN X platform, and accuracy, AUC score, on OTB-2013 dataset [30], of all compared trackers. R-MDNet indicates the retrained MDNet on ILSVRC dataset [23]. See text for more details.

*Corresponding author. Tel: (+86)-22-27406538. This work is supported by NSFC 61671325, 61572354, 61672376.

reality, could be very challenging. Generally, the essential problem is how to build a tracker that can tolerate target appearance variation, exclude background interference, while maintaining real-time tracking response.

There are two major dominant visual tracking strategies [19]. The first one adopts the classical classification & updating pipeline, which maintains an online updated classifier [29, 13] or object appearance model [34] to optimally select the most probable candidate sample as the tracked object at a coming frame [18, 30]. Recently, such classification & updating tracking scheme has been developed via online fine-tuned deep networks, thus has achieved much better (or the best) tracking accuracy [18] either by transferring some pre-trained networks [14, 25, 11] for specific tracking tasks [28, 21, 17] or by directly learning a particular tracking network [22]. However, due to the expensiveness of satisfactorily fine-tuning a deep network, such methods are usually very slow, thus are infeasible for real-world online tracking tasks. See Fig. 1 for example. The state-of-the-art deep classification & updating trackers, e.g. MDNet [22], DeepSRDCF [6] and STCT [28], can only run at 1-2fps, although they do achieve the best accuracy.[1]

The second mature strategy is matching based tracking, which matches the candidate samples with the target template and needs not online updating. The most notable advantage of such trackers is their real-time speed [2, 15]. Recently, matching based tracking can also use deep models to boost the matching generalization power [26, 2, 15, 4]. They are supposed to learn a general matching function to tolerate object online changes, while preserving real-time response ability. A recent successful model is Siamese network, SiamFC [2], which achieves promising tracking accuracy and beyond real-time speed. However, matching based tracking inherently lacks the important online adaptability, thus cannot capture the temporal variations of objects, backgrounds or imaging conditions well. This makes them still have a big accuracy gap compared to the classification & updating based trackers. As shown in Fig. 1, when similar objects coexist in the target neighborhood or the object changes significantly, matching based trackers are prone to fail, because such factors may easily disturb the pre-learned matching model, even for the state-of-the-art SiamFC [2]. A naive solution to adapt target appearance variation is to replace the target template with the tracking results obtained from previous frames [15]. But, tracking results cannot be always correct. Inevitable tracking errors could easily make this naive adaptation strategy deviate from the target object.

In this paper, we show that reliable online adaptation can be realized for matching based tracking. Specifically, we propose *dynamic Siamese network*, i.e. DSiam, with a fast *general transformation learning* model that enables effective online learning of *target appearance variation* and *background suppression* from previous frames. Since the transformation learning can be rapidly solved in FFT-domain closed form, besides the effective online adaptation abilities, it is very fast and indeed serves as a single network layer, thus can be jointly fine-tuned with the whole network. Our second contribution is *elementwise multi-layer fusion*, which adaptively integrates the multi-level deep features of DSiam network. Third, beyond most matching based trackers whose matching models are trained on image pairs, we develop a complete *joint training* scheme for the proposed DSiam network, which can be trained as a whole directly on labeled video sequences. Therefore, our model can thoroughly take into account the rich spatial temporal information of moving objects within the training videos. Extensive experiments on real-world benchmark datasets validate the balanced and superior performance of our approach.

[1] MDNet was trained on sequences directly selected from the testing benchmark that may cause unfair advantages over other trackers trained on different dataset from the testing one. To avoid such bias, we retrain MDNet on ILSVRC-2015 video dataset [23]. As shown in Fig. 1, the retrained version, denoted by R-MDNet, does not get the highest accuracy.

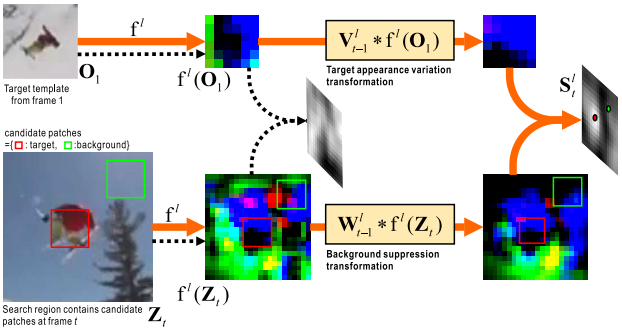


Figure 2. Basic pipeline of our DSiam network (orange line) and that of SiamFC [2] (black dashed line). $\mathrm{f}^l(\cdot)$ represents a CNN to extract the deep feature at lth layer. We add the target appearance variation ($\mathbf{V}^l_{t-1}$) and background suppression ($\mathbf{W}^l_{t-1}$) transformations for two branches respectively. Two transformations are rapidly learned from frame $t-1$. When the target at frame t (red box) is entirely different from the template $\mathbf{O}_1$, SiamFC gets a meaningless response map, within which no target can be detected. In contrast, our approach still can capture the target at frame t.

2. Related Work

Siamese network based tracking. Siamese network based trackers [2, 26] select target from candidate patches through a matching function offline learned on image pairs. The matching function is usually formulated by two-branch CNNs that share the parameters and indicate the similarity between target template $\mathbf{O}_1 \in \Re^{m\times n\times 3}$ and candidate patches cropped from a searching region $\mathbf{Z}_t \in \Re^{m_z\times n_z\times 3}$ in the tth frame. $\mathbf{O}_1$ is the target template given at the first frame. SiamFC [2] has used a fully convolutional strategy to realize this process. We briefly review SiamFC in Fig. 2 with black dashed lines and formulate it as

$$\mathbf{S}^l_t = \mathrm{corr}(\mathrm{f}^l(\mathbf{O}_1), \mathrm{f}^l(\mathbf{Z}_t)), \tag{1}$$

where $\mathbf{S}_t^l$ is a response map denoting the similarity between $\mathbf{O}_1$ and candidate patches in $\mathbf{Z}_t$; $\mathrm{f}^l(\cdot)$ represents the lth layer deep feature of some properly trained CNN model, e.g. AlexNet and VGG; $\mathrm{corr}(\cdot)$ is the correlation operation that can be replaced by other metric function, e.g. Euclidean distance used in SINT [26]. Although SiamFC can run beyond real-time, its tracking accuracy still has a big gap to state-of-the-art classification & updating trackers, due to the lack of online adaptation ability. Despite SINT achieves higher tracking accuracy, it utilizes optical flow and is much slower (about 2fps) than SiamFC. Recently, GOTURN [15] proposes to regress the target bounding box from previous frame with the Siamese network and can run at 100fps. It, however, has much lower tracking accuracy on benchmarks compared to state-of-the-art classification & updating based trackers [13, 6]. Alternatively, we propose to learn a dynamic Siamese network by introducing two online updatable transformations into the two branches respectively and then extend it to multiple layers with offline learned elementwise weight maps. Fig. 2 briefly illustrates our pipeline using single layer deep feature. Besides, via joint training on video sequences, our model achieves state-of-the-art tracking performance with real-time speed.

Deep correlation based tracking. Correlation filtering is able to realize fast tracking through circular convolution, which can be quickly solved in frequency domain, e.g. MOSSE [3], KCF [16], STC [32], DSST [5], Staple [1]. Recently, HCF [21] further extends such updating strategy to pre-trained multi-level deep features and achieves near real-time speed (about 10fps). Although we also use circular convolution to realize fast transformation and parameters learning, our model is different from previous correlation filtering trackers. First, we use circular convolution to regress a deep feature to another one and aim to capture target variation or suppress the background interference. In contrast, most correlation filtering trackers use circular convolution to regress features to a fixed Gaussian heat map and get the target location directly. Second, HCF [21] has proved the response maps from multi-layer deep features can be fused to get better performance. However, it uses artificial and fixed parameters to fuse those responses in a hierarchical way. Instead, we propose to offline learn the elementwise fusion weight maps. Third, rather than using pre-trained CNNs for other tasks as the deep features extractor, we propose to jointly train our model (both network weights and model parameters) directly on video sequences, which is much more effective for tracking problem.

3. Dynamic Siamese Network

3.1. Overview

We consider visual tracking as a joint problem of fast template matching and online transformation learning, according to the information of previous frames. Thus, beyond the original static Siamese matching model Eq. (1), we extend it into a dynamic Siamese matching process,

$$\mathbf{S}_t^l = \mathrm{corr}(\mathbf{V}_{t-1}^l * \mathrm{f}^l(\mathbf{O}_1), \mathbf{W}_{t-1}^l * \mathrm{f}^l(\mathbf{Z}_t)), \qquad (2)$$

where, as defined previously, $\mathbf{S}_t^l$ is a response map indicating the possible location of target at the tth frame; $*$ denotes circular convolution that can be fast solved in frequency domain and does not change the size of input [12]. In contrast to Eq. (1), we introduce two transformations, $\mathbf{V}_{t-1}^l$ and $\mathbf{W}_{t-1}^l$, to update the deep features of target template $\mathbf{O}_1$ and searching region $\mathbf{Z}_t$, respectively. $\mathbf{V}_{t-1}^l$ aims to encourage $\mathrm{f}^l(\mathbf{O}_1)$ being similar to $\mathrm{f}^l(\mathbf{O}_{t-1})$ and is online learned from $(t-1)$th frame by considering temporally smooth variation of the target. Thus, we denote $\mathbf{V}_{t-1}^l$ as the *target appearance variation transformation*. $\mathbf{W}_{t-1}^l$ aims to highlight the deep feature of target neighborhood regions and alleviate the interference of irrelevant background features. Hence, we denote $\mathbf{W}_{t-1}^l$ as the *background suppression transformation*. Fig. 2 illustrates the pipeline of Eq. (2). Since we add two online updatable components into the two branches of static Siamese network respectively, we call our model *dynamic Siamese network*, i.e. DSiam.

3.2. Fast transformation learning

Regularized linear regression. We use regularized linear regression (RLR) [24] to calculate $\mathbf{V}_{t-1}^l$ and $\mathbf{W}_{t-1}^l$. Generally speaking, given two tensors $\mathbf{X}$ and $\mathbf{Y}$, we aim to find an optimal linear transformation matrix $\mathbf{R}$ to make $\mathbf{X}$ being similar to $\mathbf{Y}$. Hence, we have

$$\mathbf{R} = \arg\min_{\mathbf{T}} \|\mathbf{T} * \mathbf{X} - \mathbf{Y}\|^2 + \lambda \|\mathbf{T}\|^2. \qquad (3)$$

Thanks to the desirable property of circular convolution '$*$' [24], $\mathbf{R}$ can be solved rapidly in frequency domain,

$$\mathbf{R} = \mathscr{F}^{-1}\Big(\frac{\mathscr{F}^{\star}(\mathbf{X}) \odot \mathscr{F}(\mathbf{Y})}{\mathscr{F}^{\star}(\mathbf{X}) \odot \mathscr{F}(\mathbf{X}) + \lambda}\Big), \qquad (4)$$

where $\mathscr{F}$ is discrete Fourier transformation (DFT); $\mathscr{F}^{-1}$ denotes the inverse DFT; $\star$ indicates complex-conjugate.

Target appearance variation $\mathbf{V}$. After tracking at the $(t-1)$th frame, we get the target $\mathbf{O}_{t-1}$. Rather than simply replacing target template $\mathbf{O}_1$ by $\mathbf{O}_{t-1}$, we learn the appearance variation from $\mathbf{O}_1$ to $\mathbf{O}_{t-1}$, as shown in Fig. 3. Note, we assume that the target variation is temporally smooth. So, we can apply such variation to force $\mathrm{f}^l(\mathbf{O}_1)$ being similar to $\mathrm{f}^l(\mathbf{O}_t)$, as done in Eq. (2). Specifically, we get the target appearance variation transformation $\mathbf{V}_{t-1}^l$ by

$$\mathbf{V}_{t-1}^l = \arg\min_{\mathbf{V}} \|\mathbf{V} * \mathbf{F}_1^l - \mathbf{F}_{t-1}^l\|^2 + \lambda_v \|\mathbf{V}\|^2, \qquad (5)$$

where $\mathbf{F}_1^l = \mathrm{f}^l(\mathbf{O}_1)$, $\mathbf{F}_{t-1}^l = \mathrm{f}^l(\mathbf{O}_{t-1})$; λ_v controls the regularization degree and can be learnt from labeled video

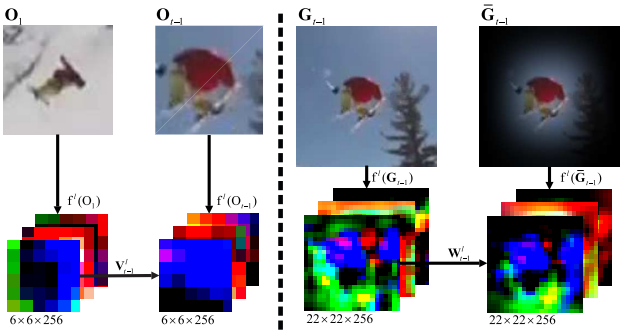


Figure 3. Two inputs of online learning target appearance variation transformation $\mathbf{V}_{t-1}^{l}$, and background suppression transformation $\mathbf{W}_{t-1}^{l}$, respectively. See text for more details.

sequences by joint training (as elaborated in section 3.5). From Eq. (4), we can efficiently obtain $\mathbf{V}_{t-1}^{l}$ by

$$\mathbf{V}_{t-1}^{l} = \mathscr{F}^{-1}\Big(\frac{\mathscr{F}^{\star}(\mathbf{F}_{1}^{l}) \odot \mathscr{F}(\mathbf{F}_{t-1}^{l})}{\mathscr{F}^{\star}(\mathbf{F}_{1}^{l}) \odot \mathscr{F}(\mathbf{F}_{1}^{l}) + \lambda_{v}}\Big). \tag{6}$$

Background suppression W. At frame t, we just want to select a candidate that has highest similarity with the transformed target template. Hence, alleviating the interference of candidates from background will help to further improve tracking accuracy. To this end, we propose to learn a transformation $\mathbf{W}_{t-1}^{l}$ that can suppress the deep features of background regions. Specifically, after tracking at the $(t-1)$th frame, we have the target location and can crop image $\mathbf{I}_{t-1}$ to region $\mathbf{G}_{t-1}$ centering at the target location and with the same size of searching region $\mathbf{Z}_{t-1}$. As illustrated by Fig. 3, we then multiply $\mathbf{G}_{t-1}$ with a Gaussian weight map and get $\bar{\mathbf{G}}_{t-1}$ to properly highlight the foreground regions. We need to learn $\mathbf{W}_{t-1}^{l}$ that encourages the deep feature of $\mathbf{G}_{t-1}$ being similar to that of $\bar{\mathbf{G}}_{t-1}$, i.e.

$$\mathbf{W}_{t-1}^{l} = \arg\min_{\mathbf{W}} \|\mathbf{W} * \mathbf{F}_{\mathrm{G}_{t-1}}^{l} - \mathbf{F}_{\bar{\mathrm{G}}_{t-1}}^{l}\|^{2} + \lambda_{w}\|\mathbf{W}\|^{2}, \tag{7}$$

where $\mathbf{F}_{\mathrm{G}_{t-1}}^{l} = \mathrm{f}^{l}(\mathbf{G}_{t-1})$, $\mathbf{F}_{\bar{\mathrm{G}}_{t-1}}^{l} = \mathrm{f}^{l}(\bar{\mathbf{G}}_{t-1})$. Similarly, through Eq. (4), we have

$$\mathbf{W}_{t-1}^{l} = \mathscr{F}^{-1}\Big(\frac{\mathscr{F}^{\star}(\mathbf{F}_{\mathrm{G}_{t-1}}^{l}) \odot \mathscr{F}(\mathbf{F}_{\bar{\mathrm{G}}_{t-1}}^{l})}{\mathscr{F}^{\star}(\mathbf{F}_{\mathrm{G}_{t-1}}^{l}) \odot \mathscr{F}(\mathbf{F}_{\mathrm{G}_{t-1}}^{l}) + \lambda_{w}}\Big), \tag{8}$$

By online learning the target variation and background suppression transformations $\mathbf{V}$ and $\mathbf{W}$, our DSiam model enables static Siamese network [2] with valuable online adaptation ability, which results in much better tracking accuracy and acceptable real-time speed. Besides, beyond previous trackers using manually set parameters, our model parameters, λ_v and λ_w, can all be learned by joint training.

3.3. Elementwise multi-layer fusion

Naturally, the DSiam model, Eq. (2), can be further extended to use multi-layer deep features. In contrast to

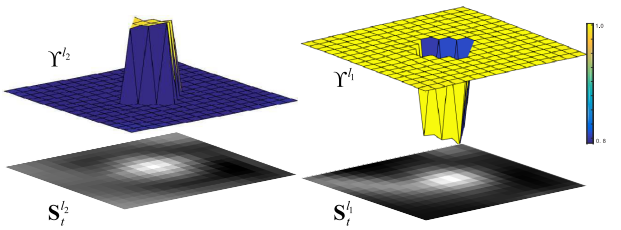


Figure 4. Real examples of offline learned weight maps $\boldsymbol{\Upsilon}^{l_1}$ and $\boldsymbol{\Upsilon}^{l_2}$, for the correlation response maps $\mathbf{S}_{t}^{l_1}$ and $\mathbf{S}_{t}^{l_2}$, from layer l_1 and l_2 ($l_1 = 5$, $l_2 = 4$) of AlexNet. Note, the response map of deeper layer l_1 has higher weights in periphery and lower weights at central part within the searching region. That is, when the target is near the center of the searching region, deeper layer features help to remove the background interference and shallower layer features are good at getting precise localization of the target; while if the target lies in periphery of the searching region, only deeper layer features are effective to determine the target location. Hence, such offline trained elementwise fusion truly reflects the complementary role of response maps from different layers, thus is helpful to obtain better target localization ability (see section 4.3).

HCF [21], we adopt a more general elementwise fusion strategy. Specifically, we can use Eq. (2) to produce $|\mathcal{L}|$ response maps $\{\mathbf{S}_{t}^{l} | l \in \mathcal{L}\}$ with multi-level features of some deep feature network. For the output $\mathbf{S}_{t}^{l} \in \Re^{m_{\mathrm{s}} \times n_{\mathrm{s}}}$ of each layer l, we set an elementwise weight map $\boldsymbol{\Upsilon}^{l} \in \Re^{m_{\mathrm{s}} \times n_{\mathrm{s}}}$ and force $\sum_{l \in \mathcal{L}} \boldsymbol{\Upsilon}^{l} = \mathbf{1}_{m_{\mathrm{s}} \times n_{\mathrm{s}}}$. The offline learning of $\boldsymbol{\Upsilon}^{l}$ is elaborated in section 3.5. Then, we can get our final response map

$$\mathbf{S}_{t} = \sum_{l \in \mathcal{L}} \boldsymbol{\Upsilon}^{l} \odot \mathbf{S}_{t}^{l}, \tag{9}$$

where $\odot$ denotes the elementwise multiplication. With Eq. (9), we have two advantages over HCF [21]: 1) elementwise fusion is much more effective that allows spatially-variant integration; 2) the weight maps can be offline learned, instead of artificially setting. See Fig. 4 for an example of two real offline learned fusion weight maps.

3.4. Network architecture

Combining Eqs. (2), (6) and (8), we get the *dynamic Siamese network* (DSiam) using single-layer deep feature, whose network architecture is shown by Fig. 5. The DSiam network can be further extended to a multi-layer version DSiamM using elementwise fusion in Eq. (9). Specifically, $\mathrm{f}^{l}(\cdot)$ denotes the deep feature of the lth layer of some proper CNN model, like VGG and AlexNet. We then introduce two new layers, circular convolution ('CirConv') and regularized linear regression ('RLR'), to formulate the fast transformation and learning of $\mathbf{V}_{t-1}^{l}$ and $\mathbf{W}_{t-1}^{l}$, Eq. (6) and (8), into a unified network representation. To make the DSiam and DSiamM architecture directly trainable on labeled video sequences rather than image pairs, we further use a 'Crop' layer to get $\mathbf{Z}_{t}$, $\mathbf{O}_{t-1}$, $\mathbf{G}_{t-1}$ and $\bar{\mathbf{G}}_{t-1}$ according to the response map $\mathbf{S}_{t-1}^{l}$. This makes the training loss can be effectively back-propagated from the last frame

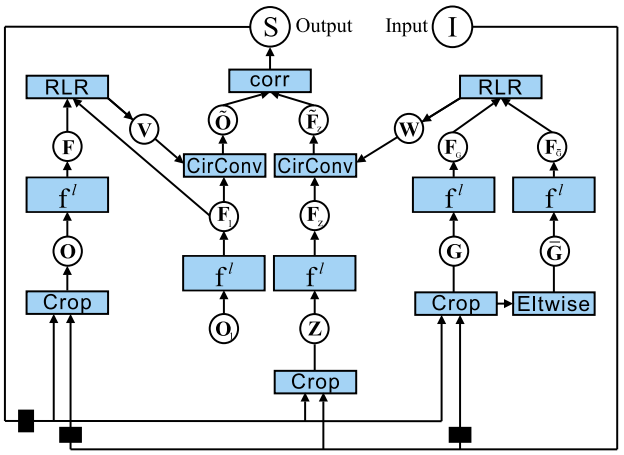


Figure 5. Network architecture of our dynamic Siamese network using single-layer deep feature $\mathrm{f}^l(\cdot)$. 'RLR' layer represents process of learning $\mathbf{V}$ and $\mathbf{W}$ defined in Eq. (6) and (8). 'CirConv' layer denotes the circular convolution '*' in Eq. (2) that leads to $\tilde{\mathbf{O}} = \mathbf{V} * \mathrm{f}^l(\mathbf{O}_1)$ and $\tilde{\mathbf{F}}_z = \mathbf{W} * \mathrm{f}^l(\mathbf{Z})$. 'Eltwise' layer aims to multiply its input with a weight map and is used in section 3.3. 'Crop' is to get regions in image $\mathbf{I}$ according to the position of maximum value in response map $\mathbf{S}$. The black block denotes a delay operation, thus we omit time index t and $t-1$ in this figure.

to the first one. Besides, an 'Elewise' layer is used to perform elementwise multiplication between $\mathbf{G}$ and a Gaussian weight map to generate $\bar{\mathbf{G}}$ (see Fig. 3 for an example). With this architecture, we can not only train the parameters of deep feature network f^l, but also can learn the elementwise weight maps and regularization parameters λ_v and λ_w of 'RLR'. As a result, the DSiam and DSiamM architecture truly allows to train a better online updatable tracker instead of just finding a good matching function [26, 2].

3.5. Joint training

To capture the rich spatial temporal information of moving objects and to learn all parameters offline, our DSiam network can be jointly trained on labeled video sequences rather than image pairs.[2] To this end, within the forward process, given a video sequence with N frames, $\{\mathbf{I}_t|t = 1, ..., N\}$, we track the target indicated at the first frame via the network architecture defined in Fig. 5. Specifically, we can get N response maps $\{\mathbf{S}_t|t = 1, ..., N\}$ that represent the tracking results at each frame. Meanwhile, we have N ground truth maps $\{\mathbf{J}_t|t = 1, ..., N\}$ with the same size of $\mathbf{S}_t$, indicating the true target location and using label 1 to denote the target and -1 to denote background. Hence, we can define the logistic loss function at each frame as

$$L_t = \frac{1}{|\mathbf{S}_t|}\log(1 + \exp(-\mathbf{S}_t \odot \mathbf{J}_t)), \tag{10}$$

[2]Classically, a small number of important model parameters can be roughly tuned by generic optimization, such as differential evolution [10]. In this paper, we propose to represent all model parameters, λ_v and λ_w, as network weights, and to learn them by gradient backpropagation.

where $|\mathbf{S}_t|$ is the size of $\mathbf{S}_t$. The total loss for the whole video is $L = \sum_{t=1}^{N} L_t$. By backpropagation through time (BPTT), we can propagate the loss to all parameters of our DSiam network, including elementwise weight maps, two RLR layers and the regularization parameters λ_v and λ_w. Unlike previous trackers, DSiam network contains two new layers, 'RLR' and 'CirConv'. To make our network trainable with BPTT and Stochastic Gradient Descent (SGD), we must get the gradient of L_t w.r.t. all parameters for these two new layers. As shown in Fig. 5, given $\nabla_{\tilde{\mathbf{O}}} L_t$, we should calculate $\nabla_{\mathbf{F}} L_t$, $\nabla_{\mathbf{F}_1} L_t$ and $\nabla_{\lambda_v} L_t$ through the left 'CirConv' and 'RLR' layers to ensure the loss gradient can be effectively propagated to f^l. Hence, we first propagate $\nabla_{\tilde{\mathbf{O}}} L_t$ to $\nabla_{\mathbf{V}} L_t$ and have

$$\nabla_{\mathbf{V}} L_t = \mathscr{F}^{-1}(\hat{\mathbf{F}}_1 \odot \hat{\nabla}_{\tilde{\mathbf{O}}} L_t). \tag{11}$$

where '$\hat{}$' denotes the Fourier transformation. From $\nabla_{\mathbf{V}} L_t$, we can then calculate $\nabla_{\mathbf{F}} L_t$ and $\nabla_{\lambda_v} L_t$ by

$$\nabla_{\mathbf{F}} L_t = \mathscr{F}^{-1}(\mathbf{U} \odot \hat{\mathbf{F}}_1^{\star} \odot \hat{\nabla}_{\mathbf{V}} L_t), \tag{12}$$

$$\nabla_{\lambda_v} L_t = \mathscr{F}^{-1}(-\mathbf{U}^2 \odot \hat{\mathbf{F}}_1^{\star} \odot \hat{\mathbf{F}})^{\mathrm{T}} \nabla_{\mathbf{V}} L_t, \tag{13}$$

$$\mathbf{U} = (\hat{\mathbf{F}}_1^{\star} \odot \hat{\mathbf{F}}_1 + \lambda_v)^{-1}. \tag{14}$$

$\nabla_{\mathbf{F}_1} L_t$ can be also derived from $\nabla_{\mathbf{V}} L_t$ and $\nabla_{\tilde{\mathbf{O}}} L_t$

$$\begin{aligned}\nabla_{\mathbf{F}_1} L_t = {} & \mathbf{E}(-2\mathbf{U}^2 \odot (\hat{\mathbf{F}}_1^{\star})^2 \odot \hat{\mathbf{F}})^{\mathrm{T}} \mathbf{E}^{\mathrm{H}} \nabla_{\mathbf{V}} L_t \\ & + \mathscr{F}^{-1}(\hat{\mathbf{V}} \odot \hat{\nabla}_{\tilde{\mathbf{O}}} L_t),\end{aligned} \tag{15}$$

where $\mathbf{E}$ is the discrete Fourier transformation matrix. The above process can also be used to calculate $\nabla_{\mathbf{F}_Z} L_t$, $\nabla_{\mathbf{F}_{\bar{G}}} L_t$, $\nabla_{\mathbf{F}_{\bar{G}}} L_t$ and $\nabla_{\lambda_w} L_t$ from $\nabla_{\tilde{\mathbf{F}}_{\mathbf{Z}}} L_t$. For elementwise multi-layer fusion Eq. (9), we can get $\nabla_{\boldsymbol{\Upsilon}^l} L_t = \mathbf{S}_t^l \odot \nabla_{\mathbf{S}_t} L_t$ to learn $\boldsymbol{\Upsilon}^l$. See the supplementary for detailed derivations.

3.6. Implementation details and the algorithm

Compared to training on image pairs, joint training on video sequences are much more complex and challenging. Hence, we use small network as f^l and choose short videos as training data. In practice, we adopt AlexNet trained by SiamFC [2] as the initialization of f^l that contains 5 convolution layers. For multi-layer fusion, we extract the deep features of 'conv4' and 'conv5' to generate response maps and get the final response map through Eq. (9). We also show in the experiments that our method also helps the pre-trained VGG19 to get much better tracking performance.

Dataset. To avoid training and testing on the same data source, we use ILSVC-2015 to train our DSiam network and test on other benchmarks. Since ILSVC-2015 has many targets occupying the whole frame that are not common in real-world tracking tasks, we first select 1130 ILSVC-2015 video sequences according to the area occupying ratio of target, from which we randomly generate 2000 training clips, each of which contains 10 successive frames.

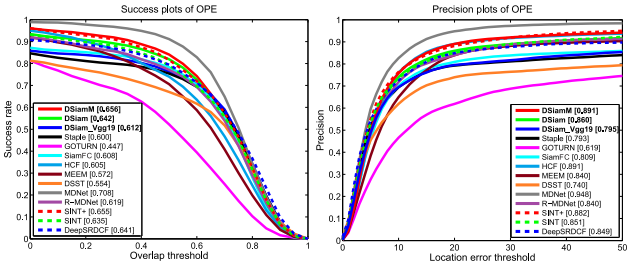


Figure 6. Success and precision plots of OPE (one pass evaluation) on OTB-2013. The numbers in the legend indicate the area-under-curve (AUC) score for success plots and the representative precisions at 20 pixels for precision plots, respectively.

Initialization. We first manually select reasonable values for λ_v and λ_w. We then update all of them via offline joint training. For elementwise fusion weight maps, we initialize the weight map of 'conv5' to be the matrix of ones and that of 'conv4' to be the matrix of zeros. We set the learning rate from 10^{-7} to 10^{-9}, weight decay 0.0005 and momentum 0.9. Our joint training is terminated at 50 iterations, which usually results in good performance.

Tracking algorithm. With the learnt dynamic Siamese network, we summary our tracking algorithm as follows: given the target location at $\mathbf{I}_1$, i.e. a bounding box $\mathbf{b}_1 \in \Re^4$, we crop the corresponding region to serve as the target template $\mathbf{O}_1$ that is slightly larger than $\mathbf{b}_1$ and centered at $\mathbf{b}_1$. We then extract the deep features of $\mathbf{O}_1$ from 'conv5' and 'conv4' layers and get $\mathbf{F}_1^4$ and $\mathbf{F}_1^5$. Before tracking, we turn off the transformations of $\mathbf{V}_0^l$ and $\mathbf{W}_0^l$ by setting them to be empty. When tracking at the tth frame, we crop search regions on three scales, i.e. $\{\mathbf{Z}_{t,s}|s \in 1,2,3\}$ centering at $\mathbf{b}_{t-1}$. Then, we get 3 response maps of $\{\mathbf{Z}_{t,s}|s \in 1,2,3\}$ via Eqs. (2) and (9). We search the maximum value among the fused response map and get its respective location and scale, which leads to $\mathbf{b}_t$. When the maximum value of fused response map is larger than 0, the tracking at current frame is successful. Then, we crop $\mathbf{I}_t$ according to $\mathbf{b}_t$ and get $\mathbf{O}_t$, $\mathbf{G}_t$, and get $\bar{\mathbf{G}}_t$ by multiplying $\mathbf{G}_t$ with a Gaussian weight map. We extract the deep features of the 3 regions and calculate $\mathbf{V}_t^{4,5}$ and $\mathbf{W}_t^{4,5}$ via Eqs. (6) and (8). We implement the proposed method in Matlab with MatConvNet toolbox [27]. Without sophisticated optimization strategies, our DSiam tracker can run at beyond real-time speed (average 45fps) on an NVIDIA TITAN X GPU.[3]

4. Experimental Results

4.1. Setup

Datasets and metrics. We evaluate our approach on two popular challenging datasets, online tracking benchmark (OTB-2013) [30] and visual object tracking 2015 benchmark (VOT-2015) [18]. OTB-2013 contains 51 real-world sequences, with 11 interference attributes,[4] and two metrics, i.e. bounding box overlap ratio and center location error. By setting a success threshold for each metric, we can get the precision and success plots, which quantitatively measure the performance of different trackers on OTB-2013 [30]. VOT-2015 has 60 sequences and re-initializes testing trackers when it misses the target. The expected average overlap considering both bounding box overlap ratio (accuracy) and the re-initialization times (robustness) serves as the major evaluation metric on VOT-2015 [18].

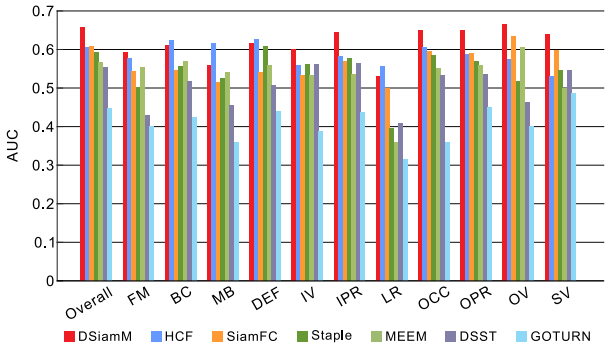


Figure 7. Specific attributes comparison of seven real-time trackers on OTB-2013 in term of success plots AUC. Our method outperforms SiamFC, Staple, MEEM, DSST and GOTURN on all 11 attributes. HCF performs better than us on attributes BC, MB, DEF and LR; while DSiamM outperforms HCF on other 7 attributes and is at least twice faster. On average, our tracker DSiamM achieves the highest accuracy among all compared real-time competitors.

Baselines. In our experiments, we choose two groups of trackers to make a thorough comparison. The first group consists of six most recent real-time trackers that can run at beyond 10fps, including DSST [5], MEEM [31], HCF [21], Staple [1], GOTURN [15] and SiamFC [2] (i.e. SiamFC_3s [2]). The second group is formed by latest trackers that produce state-of-the-art accuracy but unnecessarily with real-time tracking speed, including DeepSRDCF [7, 6], MDNet [22], SINT and SINT+ [26]. Besides, MDNet uses sequences from the benchmarks to train their model, which may not be totally fair for other competitors. Thus, we retrain MDNet on ILSVRC-2015 using the same parameters and strategy of the original version. We denote the retrained MDNet as R-MDNet. We evaluate three particular variants of our approach, DSiam, DSiamM and DSiamM_Vgg19. DSiam and DSiamM use the feature network introduced in section 3.6 as f^l. Specifically, DSiam only uses layer 'conv5'; DSiamM fuses the responses of layers 'conv5' and 'conv4' with offline learned elementwise fusion weight maps; DSiamM_Vgg19 uses the pre-trained VGG19 network [25] as f^l and adopts the deep features from 'conv5-4' and 'conv4-4' layers.

[3]The binary executable of DSiam will be publicly released online.

[4]The 11 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).

Table 1. Comparative results on VOT-2015 dataset. Note, the speed below is the normalized speed generated on VOT-2015.

Trackers	Accuracy	Overlap	Speed
DSiamM	**0.5566**	**0.2927**	**4.3498**
DSiam	**0.5414**	**0.2804**	**6.4834**
MDNet	0.5610	0.3580	0.8728
R-MDNet	0.5397	0.2334	0.4724
DeepSRDCF	0.5352	0.3040	0.3811
Staple	0.5341	0.2659	10.5529
SiamFC	0.5335	0.2889	7.3950
GOTURN	0.5121	0.2035	13.3838
DSST	0.5078	0.1678	6.7108
MEEM	0.4827	0.2179	4.9031

4.2. Comparison results

OTB-2013 dataset. As shown in Fig. 6, DSiamM achieves the second best performance in both success and precision plots. Although MDNet has the highest accuracy, it runs at only 1fps and is much slower than DSiamM and DSiam. Besides, both DSiamM and DSiam are better than its retrained version, i.e. R-MDNet, with 6% and 4% relative improvement, respectively. DSiamM (DSiam) also outperforms the other two online updated deep trackers, DeepSRDCF and HCF, on AUC of success plots, with relative improvements of 2.3% (0.2%) and 8.4% (6.1%), respectively. DSiamM performs better than recent Siamese network based trackers, SINT+, SINT and SiamFC, even though SINT+ uses optical flow as an extra motion information. Although DSiamM and DSiam are slower than SiamFC, they get 7.9% and 5.6% relative improvement over SiamFC, respectively, and both have real-time speed too. Other real-time trackers, GOTURN, Staple, MEEM and DSST, are more likely to track the target with lower accuracy and robustness or may even lose the targets within longer sequences. Specifically, DSiamM (DSiam, DSiamM_Vgg19) has achieved relative improvement of 46.8% (43.6%, 36.9%), 9.3% (7%, 2%), 14.7% (12.2%, 7%) and 18.4% (15.9%, 10.5%) over GOTURN, Staple, MEEM and DSST, respectively. These results clearly verify the superior tracking effectiveness and efficiency of our approach.

Fig. 7 further compares our approach with six state-of-the-art real-time trackers, SiamFC, Staple, GOTURN, HCF, MEEM and DSST, on 11 particular attributes of OTB-2013 benchmark. Our tracker, DSiamM, outperforms SiamFC, Staple, MEEM, DSST and GOTURN on all 11 attributes. This indicates our tracker is able to perform robust tracking with high speed under variant conditions. Although DSiamM is worse than HCF on the attributes of Background Cluttered (BC), Motion Blur (MB), Deformation (DEF) and Low Resolution (LR), DSiamM is much better than HCF on the whole dataset and is at least twice faster. Besides, the promising performance DSiamM_Vgg19 also validates the generality of our model to utilize arbitrary deep features.

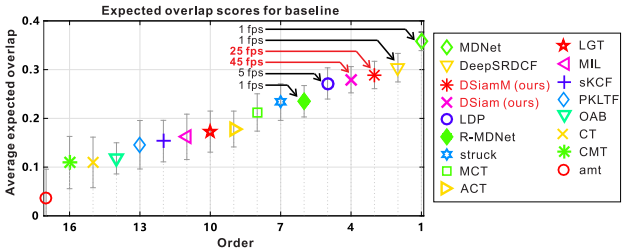


Figure 8. Expected average overlap (EAO) ranking on VOT-2015 dataset. For clarity, we only show 15 trackers in this figure.

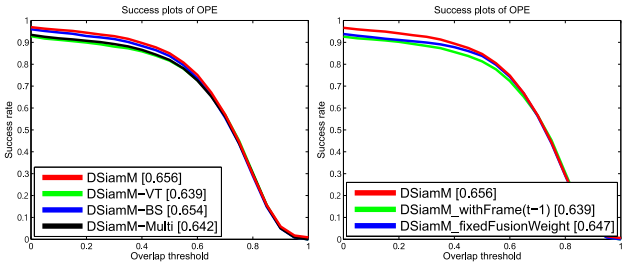


Figure 9. Left subfigure shows the comparison of three variants of DSiamM. DSiamM-VT, DSiamM-BS and DSiamM-Multi denote the trackers whose target appearance variation transformation, background suppression transformation and elementwise multi-layer fusion component are removed from DSiamM, respectively. Right subfigure compares DSiamM with another two variants, DSiamM_withFrame(t-1) simply updating the target template $\mathbf{O}_1$ to $\mathbf{O}_{t-1}$ and DSiamM_fixedFusionWeight using fixed fusion weights to fuse multi-layer tracking response maps.

VOT-2015 dataset. We show the comparative results on VOT-2015 dataset in Fig. 8 and Table 1. In Table 1, we compare our trackers with eight state-of-the-art competitors. Although DSiamM has lower overlap score than MDNet and DeepSRDCF, it runs much faster than them in term of normalized speed. Additionally, DSiamM does much better than R-MDNet that corrects the unfairly biased training advantage of MDNet. Although GOTURN, Staple and SiamFC are faster, our DSiamM tracker gets much higher accuracy than them and can also run at real-time. These results demonstrate that DSiamM is able to achieve more balanced tracking performance in terms of reliable accuracy and real-time speed. Fig. 8 shows the EAO ranking of all compared trackers in VOT-2015 challenge. The proposed DSiamM and DSiam are the top 3 and 4 in term of average overlap with 45fps and 25fps tracking speed, respectively. Besides, we can also see that DSiamM obtains apparently better average overlap than DSiam, since with elementwise multi-layer fusion, DSiamM can gather much more useful information about the moving target.

4.3. Discussion

Contributions of specific algorithmic components. The proposed DSiamM tracker has three important components, target appearance variation transformation ('VT'), background suppression transformation ('BS') and elemen-

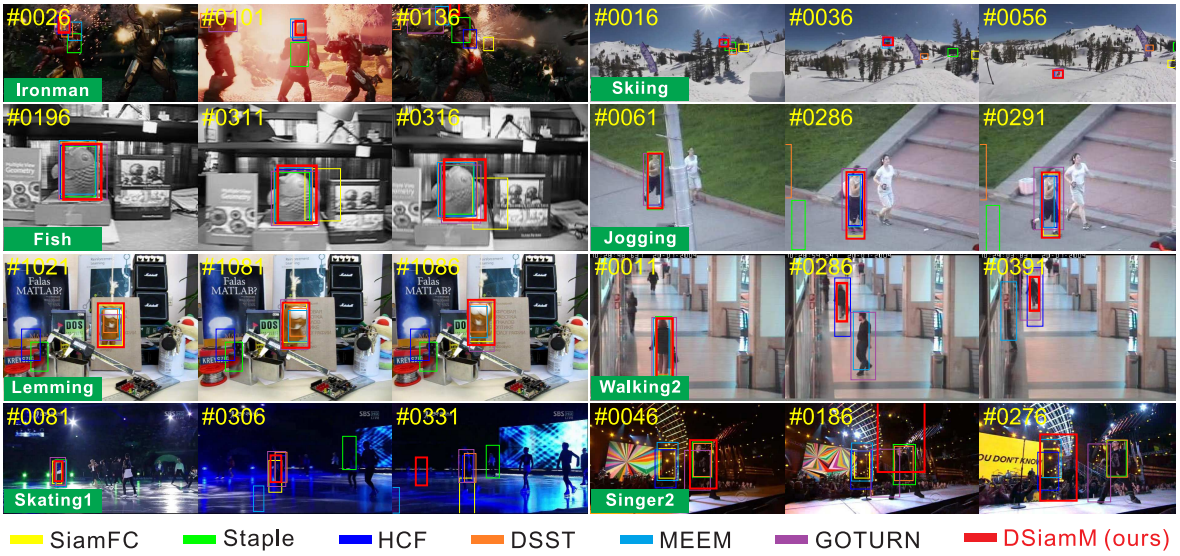


Figure 10. Tracking results of 8 typical video sequences, using our tracker and 6 real-time trackers. The 4th row shows two failure cases.

twise multi-layer fusion ('Multi'). We evaluate their concrete contributions in DSiamM by removing each one and checking the performance of degraded trackers on OTB-2013. As shown in the left subfigure of Fig. 9, the tracking accuracy decreases if we remove any component from DSiamM. Hence, all three components of DSiamM make positive contributions. Specifically, the first component 'VT' contributes the most. The third component 'Multi' also plays an important role by using multi-layers deep features.

Online adaptation strategy. Instead of the proposed target appearance variation transformation learning, a naive strategy is to simply update the target template $\mathbf{O}_1$ by $\mathbf{O}_{t-1}$. As shown in the right subfigure of Fig. 9, such simple strategy cannot lead to the best performance, since it is easier to be affected by inevitable tracking errors that may corrupt the deep features of $\mathbf{O}_{t-1}$. On the contrary, we always use the ground truth target template $\mathbf{O}_1$ and learn the temporal variation transformation by regularized linear regression, which can partially correct the influence of tracking error. Besides, we do not store the learned variation transformation that eliminates the risk of cumulative error.

Elementwise fusion vs. fixed fusion weight. An alternative fusion solution is to artificially select some proper combination weights for multi-layer response maps through generic optimization [10] or exhaustive testing. Then, all positions in a response map have the same weight. We compare our elementwise multi-layer fusion with this simpler fusion strategy. As shown in the right subfigure of Fig. 9, elementwise fusion indeed gets better tracking performance.

Failure case analysis. We show two typical failure cases of DSiamM that loses the target in the 4th row of Fig 10. DSiamM fails in these two cases mainly because the environmental illumination changes significantly and our online target variation or background suppression transformations learning could not handle such large sudden changes satisfactorily. Considering more effective features, e.g. HOG for plain images or SPHORB [35] for spherical ones, may help to alleviate this problem to some extent.

5. Conclusion

This paper has proposed dynamic Siamese network (DSiam) for visual object tracking, aiming to provide reliable online adaptation ability, while maintaining real-time tracking speed. Compared to existing competitors, our approach has three major advantages. First, stemming from state-of-the-art Siamese networks [2], our DSiam model is equipped with reliable online adaptation capabilities to the temporal variations of both foreground and background, without harming real-time response ability, thus leads to superiorly balanced tracking performance on real-world datasets. Second, our DSiam model can work on multi-level deep features, the outputs of which can be adaptively integrated through a particular elementwise fusion layer. Third, unlike most matching based trackers whose core matching models are basically trained on image pairs, our DSiam network can be jointly trained as a whole model, directly on labeled video sequences, thus can more satisfactorily capture the rich spatial temporal information of moving objects. Besides, thanks to the proposed joint training, all parameters of our model can be offline learned by backpropagation.

In the future, we plan to explore the possibility of online regressing more detailed parameters of a moving object, e.g. its scale, major orientation, aspect ratio, even tight silhouette, using the proposed dynamic Siamese network, and to further accelerate the process by superpixel representation [20, 33]. We are also interested in studying online video object segmentation via dynamic Siamese network by properly integrating with classical random field models [8, 9].

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr. Staple: Complementary learners for real-time tracking. In *CVPR*, 2016.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *arXiv preprint arXiv:1606.09549*, 2016.

[3] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010.

[4] K. Chen and W. Tao. Once for all: A two-flow convolutional neural network for visual tracking. In *arxiv:1604.07507*, 2016.

[5] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014.

[6] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCVW*, 2015.

[7] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015.

[8] W. Feng, J. Jia, and Z.-Q. Liu. Self-validated labeling of Markov random fields for image segmentation. *IEEE TPAMI*, 32(10):1871–1887, 2010.

[9] W. Feng and Z.-Q. Liu. Region-level image authentication using Bayesian structural content abstraction. *IEEE TIP*, 17(12):2413–2424, 2008.

[10] W. Feng, X. Yin, Y. Zhang, and L. Xie. NestDE: Generic parameters tuning for automatic story segmentation. *Soft Computing*, 19(1):61–70, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] R. M. Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends in Communications and Information Theory*, 2(3):155–239, 2006.

[13] Q. Guo, W. Feng, C. Zhou, C.-M. Pun, and B. Wu. Structure-regularized compressive tracking with online data-driven sampling. *IEEE TIP*, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE TPAMI*, 37(3):583–596, 2015.

[17] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, 2015.

[18] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, A. Gupta, A. Bibi, A. Lukezic, A. Garcia-Martin, A. Saffari, A. Petrosino, and A. S. Montero. The visual object tracking VOT2015 challenge results. In *ICCVW*, 2015.

[19] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Cehovin. A novel performance evaluation methodology for single-target trackers. *IEEE TPAMI*, 38(11):2137–2155, 2016.

[20] L. Li, W. Feng, L. Wan, and J. Zhang. Maximum cohesive grid of superpixels for fast object localization. In *CVPR*, 2013.

[21] C. Ma, J. B. Huang, X. Yang, and M. H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015.

[22] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[24] B. Schölkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, 2001.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[26] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[27] A. Vedaldi and K. Lenc. MatConvNet: Convolutional neural networks for Matlab. In *ACM MM*, 2015.

[28] L. Wang, W. Ouyang, X. Wang, and H. Lu. STCT: Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016.

[29] N. Wang, J. Shi, D. Y. Yeung, and J. Jia. Understanding and diagnosing visual tracking systems. In *ICCV*, 2015.

[30] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: a benchmark. In *CVPR*, 2013.

[31] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014.

[32] K. Zhang, L. Zhang, M.-H. Yang, and D. Zhang. Fast tracking via spatio-temporal context learning. In *ECCV*, 2014.

[33] S. Zhang, W. Feng, J. Zhang, and C.-M. Pun. Bag of squares: A reliable model of measuring superpixel similarity. In *ICME*, 2014.

[34] T. Zhang, S. Liu, N. Ahuja, M.-H. Yang, and B. Ghanem. Robust visual tracking via consistent low-rank sparse learning. *IJCV*, 111(2):171–190, 2015.

[35] Q. Zhao, W. Feng, L. Wan, and J.-W. Zhang. SPHORB: A fast and robust binary feature on the sphere. *IJCV*, 113(2):143–159, 2015.