Received: 1 May 2020 | Revised: 22 October 2020 | Accepted: 17 November 2020

IET Computer Vision

DOI: 10.1049/cvi2.12040

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

Object tracking using temporally matching filters

Brendan Robeson | **Mohammadreza Javanmardi** | **Xiaojun Qi**

Department of Computer Science, Utah State University, Logan, UT, USA

Correspondence

Brendan Robeson, Department of Computer Science, Utah State University, Logan, UT, USA.
Email: brendan.robeson@aggiemail.usu.edu

Funding information

Space Dynamics Laboratory, Grant/Award Number: n/a

Abstract

One of the primary challenges of visual tracking is the variable appearance of the target object. As tracking proceeds, the target object can change its appearance due to illumination changes, rotations, deformations etc. Modern trackers incorporate online updating to learn how the target changes over time. However, they do not use the history of target appearance. To address this shortcoming, we uniquely use domain adaptation with the target appearance history to efficiently learn a temporally matching filter (TMF) during online updating. This TMF emphasizes the persistent features found in different appearances of the target object. It also improves the classification accuracy of the convolutional neural network by assisting the training of the classification layers without incurring the runtime overhead of updating the convolutional layers. Extensive experimental results demonstrate that the proposed TMF-based tracker, which incorporates domain adaptation with the target appearance history, improves tracking performance on three benchmark video databases (OTB-50, OTB-100 and VOT2016) over other online learning trackers. Specifically, it improves the overlap success of VITAL and MDNet by 0.44 % and 1.03 % on the OTB-100 dataset and improves the accuracy of VITAL and MDNet by 0.55 % and 0.06 % on the VOT2016 dataset, respectively.

1 | INTRODUCTION

One of the primary challenges of visual tracking is the change in target appearance over time. For example, a car may change its colour when moving from an area of shadow to light. A bicyclist may change their orientation when performing a flip. A dancer may deform when they change the position of their limbs. A person may be partially or fully occluded when moving behind a truck. Figure 1 shows examples of occlusion and deformation. These appearance changes may have an impact on the features used by a tracking algorithm and therefore may confuse the tracker and lead to tracking failure.

Trackers built on convolutional neural networks (CNNs) are widely used to handle changes in target appearance. In a basic CNN tracker, a series of convolutional layers extract the features of the target object. Fully connected (FC) layers then classify the features from the last convolutional layer as either the target or background. Earlier, CNN trackers are static because the model does not change during tracking. Recent CNN trackers add an update phase to adjust the classification layers as the appearance of the target object changes.

Five representative CNN-based trackers including SO-DLT [1], CNN-SVM [2], MDNet [3], DSiam [4] and AVA [5] achieve improved tracking results on challenging tracking benchmarks. However, SO-DLT, CNN-SVM and MDNet have two major shortcomings. First, they only update the FC layers since it is time-consuming to learn CNN filters, which require updating millions of parameters in the convolutional layers. Second, they do not make full use of the target's known history to learn the potential future target appearance. As an example, a tracker may update its model at the moment the target has rotated in the image plane. The model may then fail to properly consider the appearance 30 frames further on when the object returns to its original orientation. To address these shortcomings, DSiam updates the target template each frame by assuming that the target variation is temporally smooth. However, this assumption is not guaranteed to hold in all cases. In addition, DSiam does not use knowledge of the past appearance of the target in the future when an appearance that the target may take on again. AVA utilizes adversarial learning to decrease the gap between the distributions of targets' appearance during tracking. However, training an adversarial

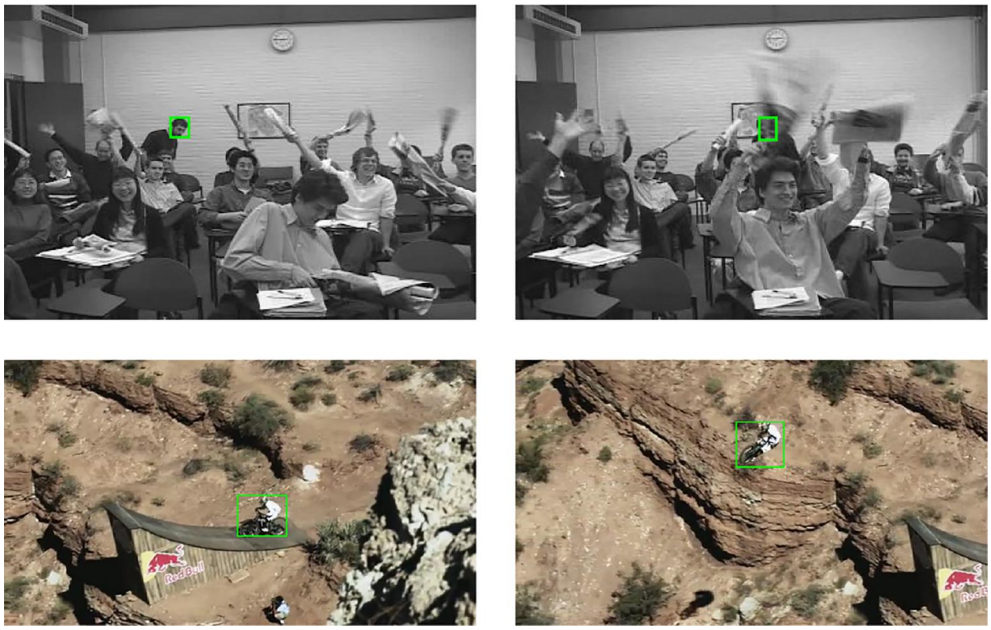

FIGURE 1 Examples of target appearance changes when people occlude the target person (top row) and the bicycle and rider deform (bottom row). The *green* boxes are the ground truth bounding boxes

network may lead to more computational time due to the increase in model parameters.

Other tracking techniques use deep features extracted from convolutional layers to improve handling of appearance variation. One representative tracker is DSAR-CF [6], which combines the weight map with a saliency map to dynamically update the weight map in the correlation filter to reflect the current target appearance. Similarly, the target history is not used to learn the range of appearances the target object can take.

Domain adaptation, also known as transfer learning, has been an ongoing research topic in the field of computer vision. It applies a model trained on one dataset for a particular task to different tasks involving different datasets [7]. Specifically, it considers the original dataset as the source domain and the new dataset as the target domain. To date, domain adaptation has shown promising results in different areas of computer vision such as image classification and object recognition, as summarized in [8–10]. To the best of our knowledge, domain adaptation has not been applied in visual tracking.

We uniquely incorporate domain adaptation in a CNN-based tracker to learn a temporally matching filter (TMF) to capture the persistent features of the target appearance history. The proposed temporally matching filter tracker (TMFT) emphasizes the features that are consistent in the target appearance history for improved training. It also improves the generalization of the classification layers by highlighting the important features of target candidates during tracking. Our extensive experiments on three challenging benchmark datasets demonstrate that TMFT achieves results competitive with state-of-the-art trackers, while avoiding the runtime cost of updating the convolutional layers. The major contributions of the proposed TMFT are:

- Learning a TMF that highlights the persistent features of the appearance of target candidates during tracking.
- Improving the model generalization of the CNN by increasing similarities of the target candidates over time.
- Designing an end-to-end CNN that simultaneously learns the TMF for target regions and the discriminative features for target detection.
- Incorporating domain adaptation into the online update process to improve the training of the classification layers and the classification accuracy.
- Organizing the feature history of the object into source and target domains for domain adaptation.

The remainder of this study is organized as follows: Section 2 discusses recent related research in feature filters and domain adaptation. Section 3 presents the proposed TMFT from the perspectives of network architecture, filter learning, network training and implementation. Section 4 presents the experimental setup and the results on OTB-50, OTB-100 and VOT2016 tracking benchmarks. Section 5 draws the conclusion and discusses the future work.

2 | RELATED WORK

In this section, we discuss recent research that is closely related to ours. We focus on reviewing two key research techniques that are used in the proposed TMFT. They are feature filters and domain adaptation.

2.1 | Feature filters

Some trackers use feature filters, also called feature masks, as part of their learning process. Song et al. [11] develop VITAL, which uses a generative adversarial network (GAN) to generate weight masks for identifying features that remain consistent over time. VITAL achieves a tracking accuracy of 0.682 on the OTB-100 dataset by avoiding overfitting with temporary features. However, VITAL may lose features that are important, but seem temporary since the full target history is not involved in updating.

Pu et al. [12] propose a deep attentive tracking (DAT) method to use attention maps to improve tracking performance. A backpropagation step is added to calculate the attention maps to capture important features of the object. It utilizes the derivatives from the first layer to provide per-pixel importance for detecting the target. A function of the attention map is then used as part of the online update loss function. However, the attention maps are not learnt from the target history, which captures the appearance of objects over time.

Han et al. [13] learn occlusion masks to make the classification layers more robust to object occlusion. The mask has binary values, where 0 indicates the presence of occlusion and 1 indicates no occlusion. Multiplying the occlusion mask with the features leads to the simulation of occlusion by cancelling certain features. However, this method may inadvertently remove discriminative features, even if the target object is never occluded in the sequence. It also does not use the target history when learning the occlusion maps.

All these feature filter-based trackers achieve better tracking accuracy by using a mask to weigh features from a different perspective. These masks make detection-based trackers more robust to appearance changes. However, they do not incorporate the feature history to learn discriminative features to represent the changing appearances of the object over time.

2.2 | Domain adaptation

Domain adaptation has been widely incorporated into the learning process in different computer vision tasks. Here, we review several representative domain adaptation-based methods.

Ganin and Lempitsky [9] pioneer learning by merging unsupervised domain adaptation with deep learning to recognize objects. They train a CNN to extract features that are not only the same in both source and target domains but also are discriminative for object recognition. However, the source domain model may extract features not present in the target domain since the source domain can contain classes not found in the target domain. Such features are detrimental to the target domain network as they can cause the network to incorrectly classify an object. To address this shortcoming, Cao et al. [8] develop a partial adversarial domain adaptation (PADA) method to mitigate the influence of these features by down weighting the information specific to the nonexistent classes.

Gaidon and Vig [14] develop an ODAMOT tracker by using online domain adaptation to track multiple objects of the same class. This adaptation occurs via sharing features of tracked objects since they belong to the same class. However, it does not learn the changing features resulting from the changing appearance of the object.

Two kinds of domain adaptation can be used in the updating process of the tracker. The first is to use a model trained for a different purpose, such as object recognition or image classification, to build the learner. It can take as little as 30 min to perform offline updating of the model compared to days or weeks to train a model from scratch. Representative trackers include MDNet [3] and its variants such as real-time MDNet [15] and DAT [12], where MDNet uses VGG-M as a base model and updates it with samples from a common visual tracking benchmark. The second kind of domain adaptation occurs during online updating. However, little research has been reported on applying domain adaptation to online updating during visual tracking.

3 | PROPOSED METHOD

We describe the proposed tracking algorithm in detail in this section. First, we present the CNN architecture and explain the functionality of each layer. Second, we describe the TMF and how it is learnt online with domain adaptation. Third, we describe in more detail the entire online update process and how the TMF is used in this process. Finally, we provide the implementation details.

3.1 | Network architecture

The proposed TMFT uses a CNN architecture to learn the persistent features of the target over time to improve classification accuracy. Building on the same network implemented in MDNet [3], we add two branches after the third convolutional layer. The first branch includes two new FC layers, FC7 and FC8, followed by a gradient reverse layer (GRL). The second branch contains a new filter application (FA) layer followed by FC4, FC5 and FC6 from MDNet. Figure 2 presents the proposed network, where *solid arrows* show the feed forward path and *dotted arrows* show the backpropagation path. For better interpretation, we colour the convolutional layers, which extract features, *red*. We colour the first branch, which learns the TMFs, *green*. We colour the second branch, which is responsible for classifying candidates as the target object or the background, *yellow*. We also include the output dimensions of each CNN layer.

The first branch learns the TMFs online to capture the persistent features of the object. To this end, we organize the appearance history of the object into source and target domains for domain adaptation. TMFT feeds the divided data to three convolutional layers to extract their feature maps, which are further sent forward through FC7 and FC8 to obtain the TMFs. The GRL assists with learning the TMFs by

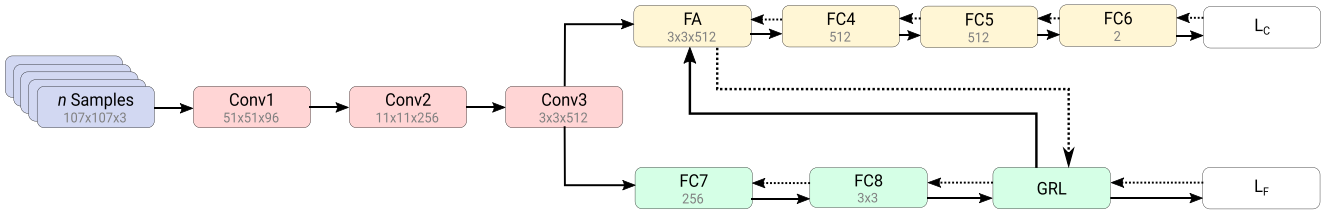

FIGURE 2 Overview of the proposed network (best viewed in colour)

negating the gradients of FC7 and FC8 during the back-propagation process. It passes the TMFs through unmodified during the feed forward process.

The second branch classifies the candidate as the target object or the background. To this end, the TMFs go through the FA layer to compute the scaled feature maps for different representations of the target object. The scaled feature maps give more weight to features present in the target object undergoing appearance changes. TMFT passes the scaled features to FC layers 4 through 6 to classify the candidate samples as the target object or the background.

3.2 | Filter learning

We use domain adaptation to learn the TMFs, which scale the feature maps generated by the third convolutional layer. Specifically, we organize the feature history of the object into source and target domain datasets. We consider features of the target object early in the history as the source domain and features of the target object later in the history as the target domain. In other words, we exclusively use the features of the target object, which are positive, to create the history.

Initially, the positive feature history contains features extracted from the first frame. During the pretraining step, TMFT collects features from 500 random samples that overlap the ground truth bounding box with an overlap ratio greater than or equal to 0.7. During tracking, features are added to the history for every frame in which the target object is successfully located. If TMFT fails to locate the target object, the feature history is not updated. Location failure occurs when the network output is below a predetermined threshold score (i.e. 0). For each such frame, features from 50 random samples, with ground truth overlap values greater than or equal to 0.7, are extracted and appended to the history. The feature history is a queue with a maximum capacity specified as a number of frames. Before features from a new frame are enqueued, features from an old frame at the head of the queue must be dequeued when the feature history reaches the maximum capacity. This is done to avoid filling memory, which would considerably slow the tracker.

This positive feature history is used to learn the TMFs required for updating the CNN. The model is updated after every 10 frames to ensure that a sufficient number of positive training features are collected for learning. We consider this update as a long-term update. When an update occurs after frame $\boldsymbol{F}_t$ where t is a multiple of 10, we have a history of features $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ that represent the changing target appearance. Here, N is the number of features in the history. Before updating the CNN, we divide the positive feature history into two disjoint subsets: source domain dataset X_S and target domain dataset X_T, where $X_S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\frac{N}{2}}\}$ and $X_T = \{\boldsymbol{x}_{\frac{N}{2}+1}, \boldsymbol{x}_{\frac{N}{2}+2}, \ldots, \boldsymbol{x}_N\}$.

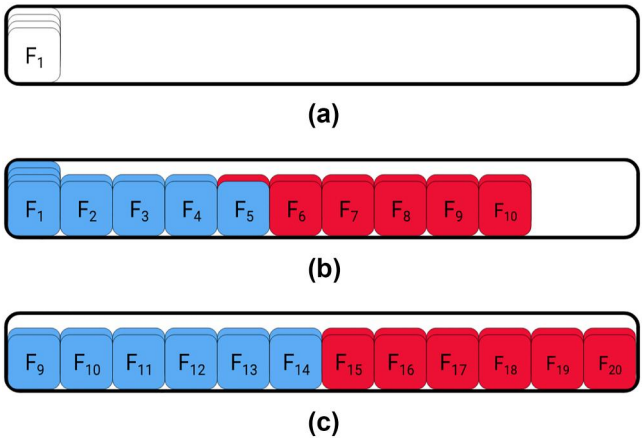

FIGURE 3 Construction of the positive feature history. (a) Positive feature history after pre-training on frame 1. (b) Positive feature history during the first long-term update after frame 10. (c) Positive feature history during the second long-term update after frame 20

Figure 3 is a simple example to illustrate the accumulated positive features in the history queue. For easy interpretation, the diagram shows four random samples extracted from the first frame, instead of 500 random samples as implemented in TMFT. It shows two random samples extracted from the successfully tracked frames, instead of 50 random samples as implemented in TMFT. The history capacity for this example is 12 frames. The source domain dataset consists of the features shown in *blue* and the target domain dataset consists of the features shown in *red*. It should be noted that the features in the two domain datasets may come from the same frame only when the features in the first frame are present in the history, as shown in Figure 3(b). Once the features in the first frame are removed from the history, the features in the source domain and the target domain are exclusively extracted from different frames, as shown in Figure 3(c).

We input features from the source and target domain datasets to the CNN to calculate their corresponding TMFs. Each of these filters is generated by FC8 and is a 3×3 matrix, whose size is dictated by the feature maps generated by the third convolutional layer, as shown in Figure 2. The values in

the TMFs are unbounded. As a result, we use a sigmoid function to transform the values to the range [0, 1].

In the proposed CNN, we use conventional binary cross-entropy (BCE) to compute the filter loss, L_f, for a single TMF $\boldsymbol{f}$ as follows:

$$L_f(\boldsymbol{f}, z_f) = -\sum_{i,j} z_f \log f_{ij} + (1 - z_f)\log\left(1 - f_{ij}\right). \quad (1)$$

This loss is a summation of the BCE for each element f_{ij} in filter $\boldsymbol{f}$, where $i \in \{1, 2, 3\}$ and $j \in \{1, 2, 3\}$ specify the filter's row number and column number, respectively. z_f is the domain label, where $z_f = 0$ for the source domain and $z_f = 1$ for the target domain. These label values follow the convention established by [9] to perform the classification task. The total filter loss is the average of the loss for all N TMFs, which is computed by:

$$L_{\mathrm{F}} = \frac{1}{N}\sum_{n=1}^{N} L_f(\boldsymbol{f}_n, z_{f_n}). \quad (2)$$

Unlike conventional training, which teaches a CNN to produce different output for input data of different classes by minimizing the loss, we use domain adaptation to learn to produce identical filters for input data of different domains by maximizing the loss in Equation (2). Since maximization problems are more difficult to solve than minimization problems, we turn the maximization problem into a minimization problem by negating the gradients [9]. Specifically, we use the GRL from PADA [8] to allow the TMF to pass through unmodified in the forward direction. During backpropagation, the GRL assists with learning the TMFs by negating the gradients of FC7 and FC8. To this end, the GRL first computes an adaptive gradient scale factor as a function of the current training iteration i as follows:

$$\mathrm{GRL}(i) = -\lambda\left(\frac{2(h-l)}{1 + \mathrm{e}^{\frac{-\alpha i}{I}}} - (h - l) + l\right), \quad (3)$$

where I is the maximum number of iterations, h and l are real positive values and respectively represent the high and low bounds of the scalar's magnitude, α is a predefined parameter to control how quickly the function reaches the minimum value $-h$, and λ is a predefined scaling factor. This adaptive gradient scale factor is in the range of (−h, −l). In the proposed CNN, we empirically determine the optimal GRL parameters and set them as follows: $h = 1$, $l = 0$, $I = 200$, $\alpha = 10$ and $\lambda = 1$. These parameters reduce Equation (3) to:

$$\mathrm{GRL}(i) = \left(1 - \frac{2}{1 + \mathrm{e}^{-\frac{i}{20}}}\right). \quad (4)$$

The GRL then negates and scales the gradients using the adaptive gradient scale factor by:

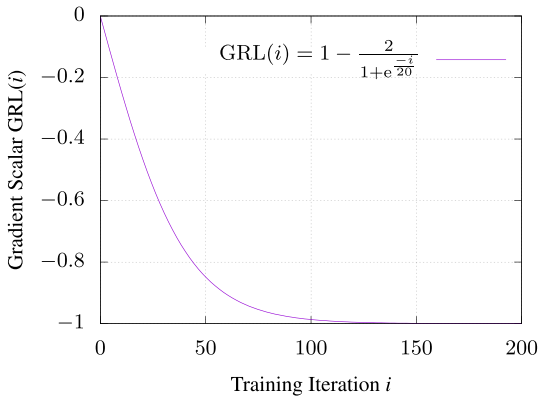

FIGURE 4 Plot of the empirically determined GRL function

$$\frac{\partial L_{\mathrm{F}}}{\partial \theta} \leftarrow \mathrm{GRL}(i)\frac{\partial L_{\mathrm{F}}}{\partial \theta}. \quad (5)$$

Figure 4 shows the empirically determined GRL function for 200 iterations. It demonstrates that the gradient scalar is reduced during the first 50 iterations and approaches an asymptote at −1 near iteration 100. The magnitude of the adaptive gradient scale factor ensures that a larger update occurs in later training iterations.

3.3 | Network training

Learning the TMFs is a crucial component of the overall online update process. This process can be summarized as (1) accumulate positive and negative features during tracking; (2) perform short-term updates when tracking fails and (3) perform regularly scheduled long-term updates.

In addition to generating the positive feature history to learn the TMFs, we also extract negative features when TMFT locates the target object. TMFT extracts features from 5000 negative samples during the pretraining step and extracts features from 200 negative samples in each tracked frame. For the pretraining step, negative samples are defined as those that have an overlap ratio with the ground truth less than or equal to 0.5. For tracked frames, the overlap ratio threshold is lowered to 0.3. These are the threshold values used in MDNet, our experimental baseline. We use the same two thresholds in TMFT to ensure fair comparison between TMFT and MDNet.

A short-term update occurs when the tracker fails to locate the object. It performs normal feed forward and backpropagation to update FC4, FC5 and FC6. The data flows from FC4 to FC6 without using TMFs. The classification loss is calculated and used to update the FC layers.

A long-term update occurs every 10 frames to learn the TMFs using the positive feature history and improve classification accuracy using both positive and negative samples. The classification learning process is similar to the short-term update process. The difference between the long-term and short-term updates is the positive inputs to FC4; during the long-term update, the inputs are the TMF scaled features. At the

beginning of a long-term update, we generate a 3×3 TMF $\boldsymbol{f}$ for each 3 × 3 × 512 feature map $\boldsymbol{x}$. The FA layer uses $\boldsymbol{f}$ to scale $\boldsymbol{x}$ by:

$$\mathrm{FA}(\boldsymbol{x}, \boldsymbol{f}) = \begin{cases} \boldsymbol{x} & \text{tracking} \\ \boldsymbol{x}_{i,j,k} \times \boldsymbol{f}_{i,j} & \text{training} \end{cases} \quad (6)$$

The subscripts i and j specify row and column positions in the filter $\boldsymbol{f}$ and the feature map $\boldsymbol{x}$, respectively. The subscript k represents the depth of $\boldsymbol{x}$. During tracking, the FA passes the feature map through to the classification layer FC4. The TMF is not used during tracking since the classification layers have already learnt which features are discriminative for different appearances of the target object. During online training, the FA uses the TMF $\boldsymbol{f}$ to perform element-wise multiplication with the feature map $\boldsymbol{x}$ at each depth plane to obtain the scaled feature map. Small TMF values de-emphasize the importance of features and large TMF values magnify the importance of features in the feature map. Layers FC4, FC5 and FC6 then use the scaled feature map to classify the sample as the target object or the background. Finally, the classification loss is calculated.

We use BCE to calculate the classification loss for a single sample. The total classification loss is the average classification loss of all samples computed by:

$$L_{\mathrm{C}} = -\frac{1}{M}\sum_{m=1}^{M} y_m \log p_m + (1 - y_m)\log(1 - p_m), \quad (7)$$

where M is the number of positive and negative samples, p_m is the probability for the mth sample to be the target object and y_m is the ground truth class label for the mth sample, with $y_m = 0$ when the mth sample is the background and $y_m = 1$ when the mth sample is the target object.

We combine the total classification loss with the total filter loss to compute the overall network loss:

$$L = L_{\mathrm{C}} + L_{\mathrm{F}}. \quad (8)$$

Since L_{F} is a constant with respect to FC4–FC6, the filter loss derivatives for those layers are 0's. In other words, layers FC4–FC6 do not contribute to the filter loss. As a result, we can sum the loss values before backpropagation.

The filter loss L_{F} is calculated only for the positive samples in the positive feature history. The classification loss is calculated for both positive and negative samples. The overall network loss seamlessly combines the two branches in the proposed CNN and ensures that training the classification branch and the filter branch occurs simultaneously. This simultaneous learning is necessary since the TMFs improve the classification results and the reduced classification errors are fed back into the CNN to learn the TMFs that better represent the persistent features. At this point, the overall network loss is backpropagated and all FC layers are updated. This backpropagation does not affect the FA layer since no weights are involved in the FA layer. It should be emphasized that the features are only scaled during long-term updates to learn the persistent features of the target object over time. The scaled feature map is not used during tracking or short-term updates.

Algorithm 1 summarizes the long-term update process. The network uses batch training, where a batch size of 32 is chosen for positive target data and a batch size of 96 is chosen for negative background data. These two batch sizes are shown to be effective in achieving good tracking results in MDNet and therefore are used in TMFT, which also facilitates fair comparison with MDNet when evaluating the tracking performance. Half of the positive batch is from the source domain and the other half is from the target domain. During each training phase, TMFT randomly permutes the features in each domain and ensures that each sample's feature map is used exactly once.

Algorithm 1 Long-term online update

Input: CNN
Positive features X^+
Negative features X^-
Output: Updated CNN
Divide X^+ into X_{S}^+ and X_{T}^+
foreach *training batch* **do**
// Filter Forwarding
Feed forward X_{S}^+ batch; output filters f_{S}
Feed forward X_{T}^+ batch; output filters f_{T}
Calculate filter loss L_{F}
//Classification Forwarding
Feed forward X_{S}^+ batch with filters f_{S}
Feed forward X_{T}^+ batch with filters f_{T}
Feed forward X^- batch
Calculate classification loss L_{C}
//Optimization
$L \leftarrow L_{\mathrm{C}} + L_{\mathrm{F}}$
Backpropagate
Update layers FC4–FC8
end

3.4 | Implementation

We implement the proposed TMFT on top of the Python implementation of MDNet [3], which serves as our experimental baseline. Layers FC7, FC8 and GRL are implemented as a separate PyTorch network.

The positive feature history capacity is 100 frames. When the history is full, the difference between the source and target ranges from 10 to 1000 frames. The last frame in the source and the first frame in the target have the smallest distance of 10 frames. The first frame in the source and the last frame in the target have the largest distance of 1000 frames. Due to the significant differences between the source and the target, domain adaption can be effectively used to learn how the target appearance varies with time. Each update phase is limited to 200 training iterations. TMFT uses stochastic gradient descent

to update the FC layers' weights. The momentum is 0.9 and the weight decay is 0.0005. The learning rate for FC6 is 0.01 and the learning rate for all the other layers is 0.001. All these parameters, except the training iterations, are set to the same values reported by Nam and Han, the authors of MDNet [3].

4 | EXPERIMENTS

We run tracking experiments using the OTB one pass evaluation protocol [16] and the VOT evaluation protocol [17] for OTB and VOT datasets, respectively. The two datasets contain different sequences and the two protocols report different evaluation measurements, which provide a more thorough evaluation of the compared trackers and help us gain more insight into the performance of each tracker. Specifically, we provide a detailed performance analysis of TMFT and its two peer trackers, MDNet and VITAL. Both TMFT and VITAL improve MDNet by adding feature filters as part of their learning process. VITAL uses a GAN to generate weight masks for identifying features that remain consistent over time. TMFT uses domain adaptation during online updating to learn a TMF that emphasises the consistent features in the target appearance history. For all experiments, we use the newly released MDNet model trained on ImageNet-Vid [18] instead of the original MDNet model trained on a subset of OTB and VOT sequences to build the proposed CNN architecture. ImageNet-Vid consists of sequences not found in VOT or OTB so any testing bias from the original MDNet is avoided in the evaluation. We run experiments on a Dell Precision Mobile Workstation with 12 CPU cores, 32 GB of RAM and an NVIDIA QuadroP1000 GPU. The GPU has 512 CUDA cores and 4 GB of RAM.

4.1 | OTB experiments

The OTB benchmark consists of two datasets: OTB-100 [19] and OTB-50 [16]. OTB-100 has 100 sequences with axis-aligned ground truth bounding box annotations. Each sequence has tags denoting 11 applicable tracking challenges including fast motion, background clutter, motion blur, deformation, illumination variation, in-plane rotation, low resolution, occlusion, out-of-plane rotation, out-of-view and scale variation. OTB-50 is a precursor to OTB-100. It consists of 50 sequences with axis-aligned ground truth bounding boxes.

For the OTB experiments, we compare TMFT with 25 state-of-the-art trackers including DSST [20], KCF [21], TGPR [22], MEEM [23], MUSTer [24], LCT [25], RSST [26], SRDCF [27], SiamFC [28], DeepSRDCF [29], ADNet [30], CFNet [31], SGLST [32], SCT [33], CNN-SVM [2], CCOT [34], ECO [35], MDNet [3], VITAL [11], CREST [36], TRACA [37], SiamRPN [38], STAPLE [39], CNT [40] and HDT [41]. We use two common performance metrics, namely, overlap success and centre error precision, to evaluate the performance of each tracker. Overlap success is a measurement of how much the tracker's bounding box overlaps with the ground truth bounding box. It is a ratio of the boxes' intersection to their union. Centre error precision is a measurement of the distance between the centre of the tracker's bounding box and the centre of the ground truth bounding box.

4.1.1 | OTB-100

Figure 5 shows overlap success and centre error precision plots for the overall OTB-100 experiments. For easy reading and clarity, we only present the top 10 trackers for each subplot. Figure 5 demonstrates that ECO achieves the best overlap success score of 0.691 and TMFT achieves the second best overlap success score of 0.685 for the OTB-100 dataset. TMFT improves VITAL (the third best tracker) and MDNet (the fourth best tracker) by 0.44 % and 1.03 % in overlap success scores, respectively. Figure 5 also illustrates the superior performance of TMFT in terms of centre error precision. TMFT achieves the best centre error precision score of 0.928 and VITAL achieves the second best centre error precision score of 0.917 for the OTB-100 dataset.

Figure 6 shows overlap success plots for the six appearance-based challenges. Specifically, TMFT consistently ranks as one of the top three trackers for these challenges. It yields the best tracking performance when the target deforms or rotates in the image plane. It outperforms its two peers when the target is occluded or undergoes out-of-plane rotations or motion blurs. It performs better than MDNet in all challenge subsets except for the fast motion subset. Overall, TMFT performs well for six appearance change-based challenges such as deformation, illumination variation, in-plane rotation, occlusion, out-of-plane rotation and out-of-view. It also performs better than MDNet and VITAL in six and four appearance change-based challenges, respectively.

Specifically, it consistently ranks as one of the top three trackers for all 11 challenge subsets except for out-of-view subset. Figure 7 shows centre error precision plots for the six appearance-based challenges. Specifically, TMFT consistently ranks as one of the top three trackers for these challenges, except out-of-view, for which TMFT ranks fourth. It achieves the best centre error precision when the target undergoes motion blur, deformation, illumination variation, in-plane rotation, out-of-plane rotation or scale variation. Overall, it ranks the best for four of six appearance change-based challenges: deformation, illumination variation, in-plane rotation and out-of-plane rotation. It outperforms MDNet for the two remaining appearance change-based challenges (i.e. occlusion and out-of-view) and outperforms VITAL for occlusion.

4.1.2 | OTB-50

Figure 8 presents the overall overlap success and centre error precision plots of the top 10 trackers for all sequences in the OTB-50 dataset. TMFT ranks fourth in terms of overlap

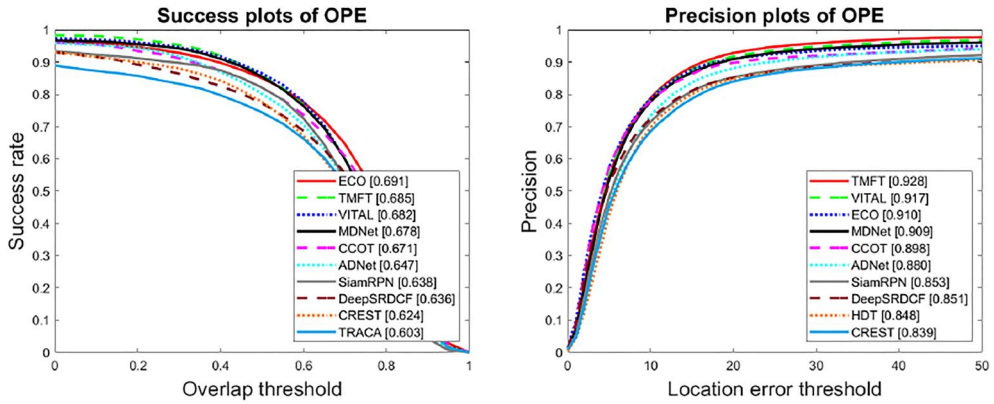

FIGURE 5 Overlap success and centre error precision plots of the top 10 trackers for the OTB-100 dataset

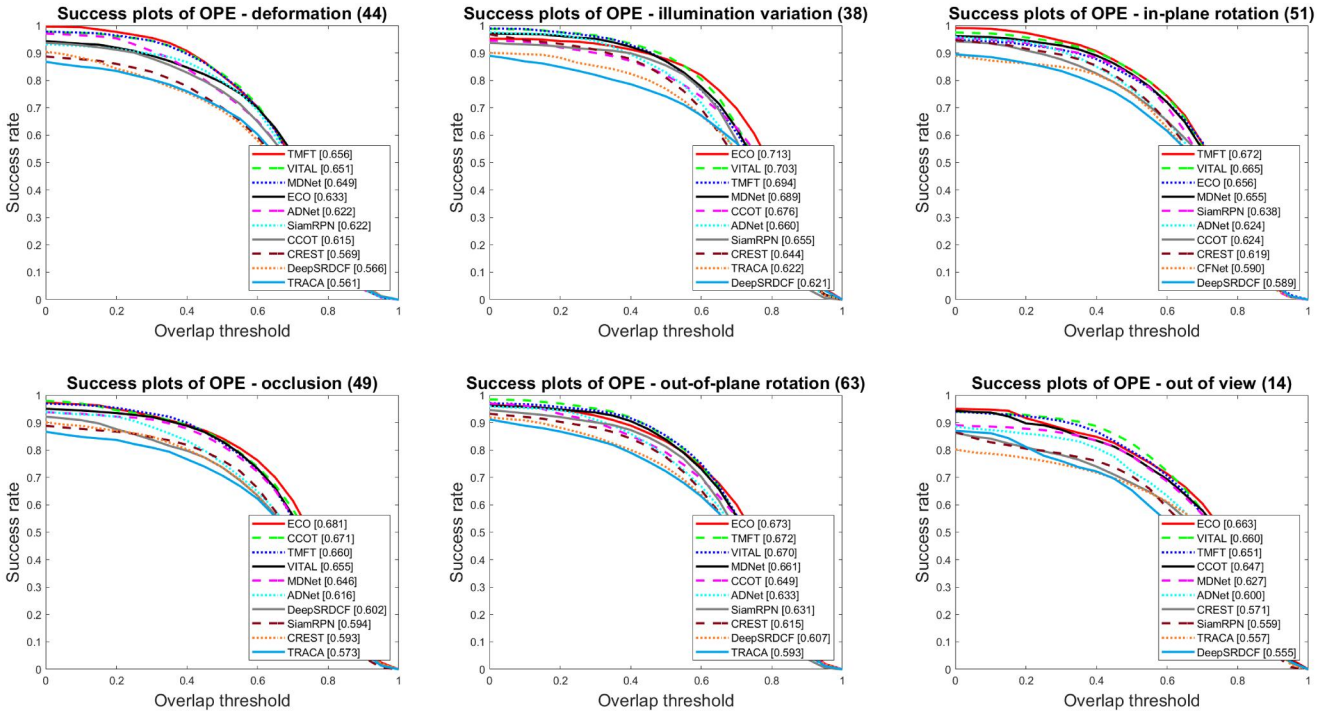

FIGURE 6 Overlap success plots of top 10 trackers for the OTB-100 dataset

success, where its score is decreased by 0.56 % when compared to VITAL (the best tracker) and by 0.28 % when compared to MDNet (the third best tracker). TMFT achieves the best centre precision of 0.953, which improves on VITAL's precision by 0.32 % and MDNet's precision by 0.53 %.

4.2 | VOT experiments

We evaluate the performance of TMFT on the VOT2016 [42] and VOT2018 [43] datasets. Each dataset consists of 60 sequences, 50 of which are shared between VOT2016 and VOT2018. The datasets also contain bounding box annotations, where bounding boxes are not axis aligned. The VOT datasets also contain challenge tags for each frame instead of challenge tags for each sequence as in the OTB dataset.

The VOT protocol uses accuracy, robustness and expected average overlap (EAO) [42] as the measurements to evaluate tracking performance. Accuracy is the overlap between the tracker's bounding box and the ground truth bounding box averaged over all frames. This metric has a value in the range [0, 1] with 1 indicating perfect tracking. Robustness is a measure of how many times the tracker loses the target object with smaller values indicating the tracker loses the target fewer times. EAO is a prediction of how well the tracker would perform on other sequences with similar attributes and of similar length as the sequences in the dataset. Similar to

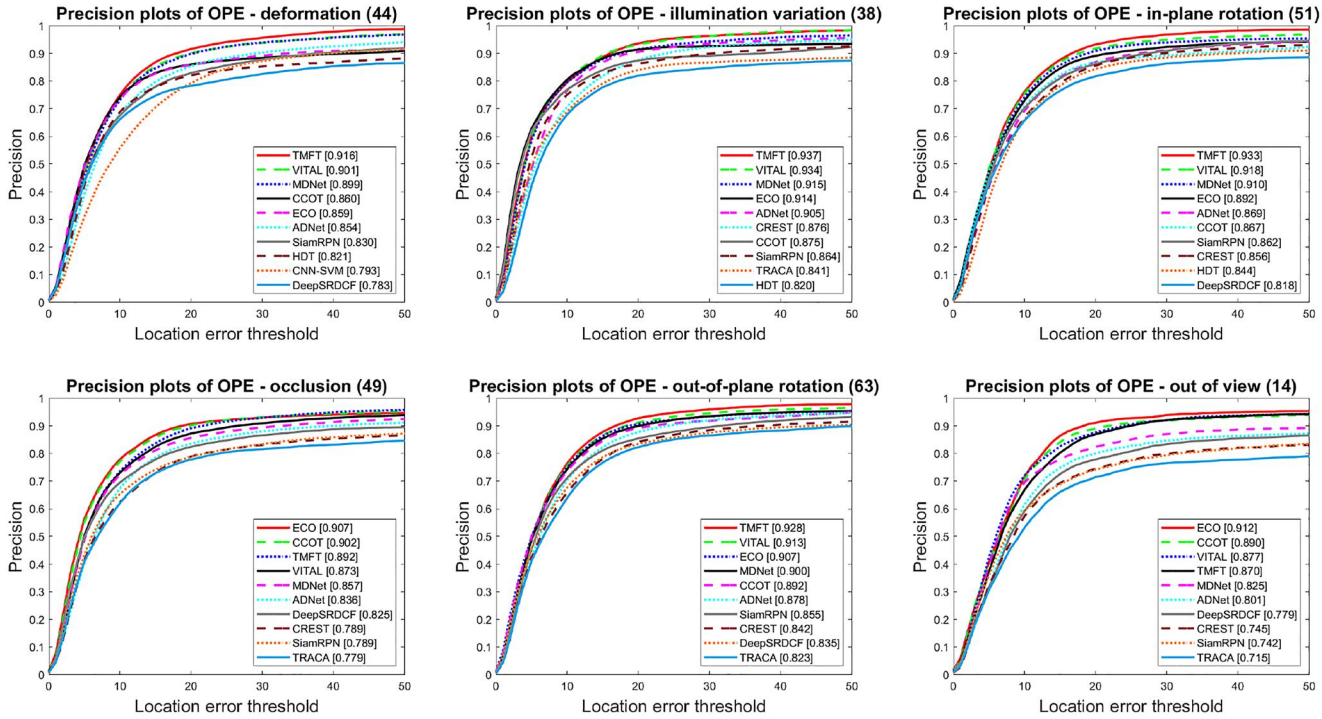

FIGURE 7 Centre error precision plots of top 10 trackers for the OTB-100 dataset

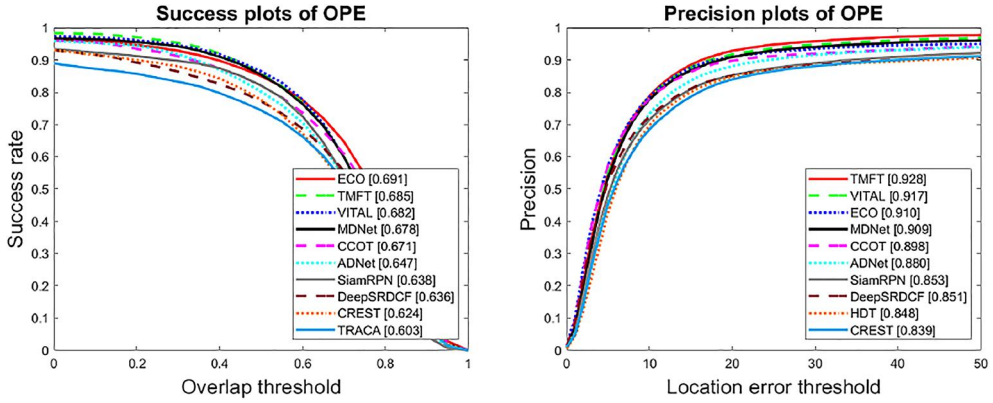

FIGURE 8 Overlap success and centre error precision plots of the top 10 trackers for the OTB-50 dataset

accuracy, EAO is an overlap ratio. However, it does not include resets when the tracker loses the target object. The EAO score will be lower than the accuracy score when a tracker loses the target object. A higher EAO score indicates better tracking performance. We refer interested readers to [44] for complete details on how EAO is calculated.

4.2.1 | VOT2016

We compare the proposed TMFT with four trackers including CCOT [34], ECO [35], MDNet [3] and VITAL [11], whose experimental results on the VOT2016 dataset are publicly available. These four compared trackers are also the top trackers for the OTB-100 dataset. Table 1 summarizes the VOT2016 tracking results using the VOT protocol. The best performance is shown in *red* and the second best performance is shown in *blue*. It is clear that TMFT outperforms all four other trackers in terms of all three measurements by achieving the highest average

TABLE 1 VOT2016 tracking results of five trackers

Tracker	Accuracy	Robustness	EAO
TMFT	0.5439	10.2333	0.3753
MDNet	0.5436	16.9333	0.2579
VITAL	0.5409	16.5000	0.3228
ECO	0.5390	10.8333	0.3738
CCOT	0.5322	14.0000	0.3294

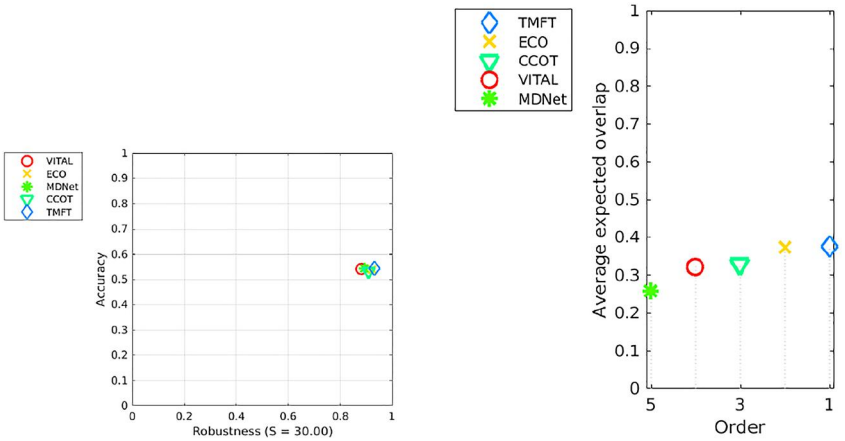

FIGURE 9 Accuracy versus robustness and EAO plots of five trackers for VOT2016

TABLE 2 VOT2018 tracking results of seven trackers

Tracker	Accuracy	Robustness	EAO
TMFT	0.534	1.507	0.227
MDNet	0.514	1.996	0.187
VITAL	0.530	1.718	0.219
ECO	0.484	0.276	0.280
CCOT	0.494	0.318	0.267
LADCF	0.503	0.159	0.389
MFT	0.505	0.140	0.385
SiamRPN	0.586	0.276	0.383

accuracy of 0.5439, the best robustness of 10.2333 and the best EAO score of 0.3753. ECO achieves the second best performance in terms of robustness and EAO and MDNet achieve the second best performance in terms of accuracy. TMFT significantly improves its two peer trackers in robustness and EAO measurements. Specifically, it improves the robustness of VITAL and MDNet by 37.98 % and 39.56 %, and the EAO score of VITAL and MDNet by 16.26 % and 45.52 %, respectively. It improves the accuracy of its peer trackers by a small margin of 0.55 % for VITAL and 0.06 % for MDNet.

Figure 9 shows the accuracy versus robustness plot and the EAO plot of the five compared trackers for the VOT2016 dataset. It is clear that TMFT is the most robust and accurate tracker among the four compared trackers. Based on the EAO measurement, TMFT is expected to outperform the other four trackers in real-world tracking scenarios.

4.2.2 | VOT2018

Table 2 summarizes the VOT2018 tracking results of TMFT and seven state-of-the-art trackers. Three of these trackers, namely, LADCF [45], MFT [46] and SiamRPN [38], are top performing trackers for the VOT2018 dataset. MDNet and VITAL are TMFT's peer trackers and ECO and CCOT are two top performing trackers for both OTB-100 and VOT2016 datasets. We use *red* to indicate the best performance and *blue* to indicate the second best performance.

It is clear that TMFT achieves the second best accuracy score of 0.534 and improves the accuracy of its two peer trackers by a small margin of 0.75 % for VITAL and 3.89 % for MDNet.

It also significantly improves its two peer trackers in both robustness and EAO measurements. Specifically, it improves the robustness of VITAL and MDNet by 12.28 % and 24.50 %, and the EAO score of VITAL and MDNet by 3.65 % and 21.39 %, respectively.

4.3 | Speed analysis

We evaluate the average tracking frequencies of TMFT and its two peers on the VOT2018 dataset to compare their tracking speed. Average tracking frequencies are 1.2054 frames per second (fps) for TMFT, 1.3458 fps for VITAL and 1.3616 fps for MDNet.

This evaluation result concurs with our expectation that TMFT should run slower than its peer trackers (MDNet and VITAL) since layers FC7 and FC8 introduce additional 2304 parameters, which are updated every 10 frames with each training iteration at the long-term update stage, to perform the domain adaptation learning task.

4.4 | Qualitative results

Figures 10–14 demonstrate sample qualitative tracking results of five compared trackers on five OTB sequences: *Bolt*, *MountainBike*, *Crowds*, *Freeman4* and *Bird1*, respectively. These sequences represent the gamut of tracking challenges identified in the OTB dataset. In each frame, a white + marks the ground truth centre. We cropped the frames to show only the target and all bounding boxes.

All five trackers are able to track the objects in *Bolt*, *MountainBike* and *Crowds*, where the object deforms in *Bolt* and *MountainBike* and the person moves from bright sun to shadow and passes through some background clutter in the form of other people in *Crowds*. Consistent features learnt

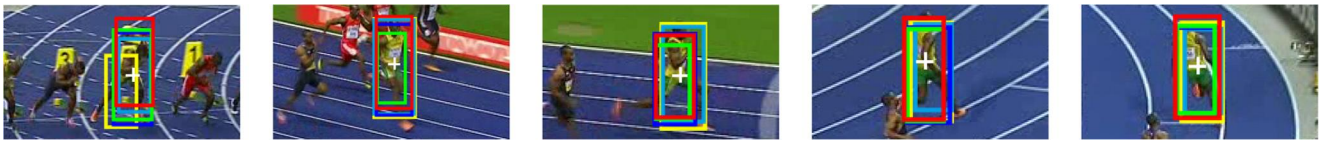

FIGURE 10 Qualitative evaluation of TMFT, CCOT [34], ECO [35], MDNet [3] and VITAL [11] on the OTB sequence, *Bolt*

FIGURE 11 Qualitative evaluation of TMFT, CCOT [34], ECO [35], MDNet [3] and VITAL [11] on the OTB sequence, *MountainBike*

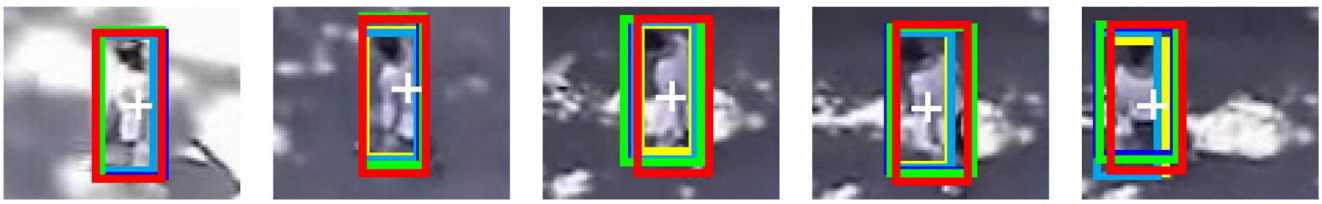

FIGURE 12 Qualitative evaluation of TMFT, CCOT [34], ECO [35], MDNet [3] and VITAL [11] on the OTB sequence, *Crowds*

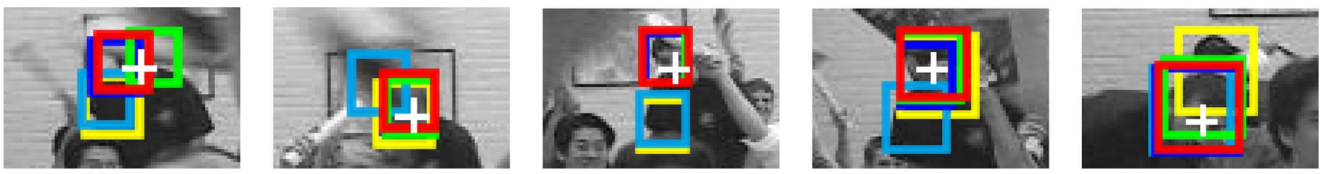

FIGURE 13 Qualitative evaluation of TMFT, CCOT [34], ECO [35], MDNet [3] and VITAL [11] on the OTB sequence, *Freeman4*

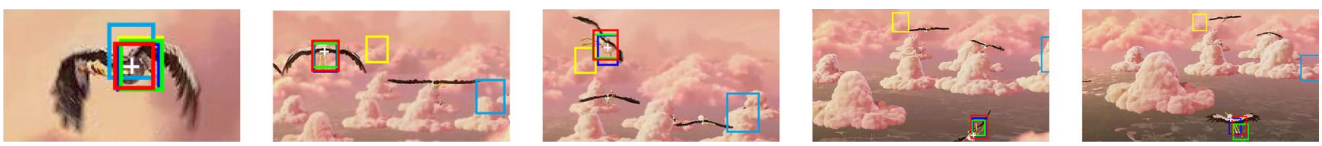

FIGURE 14 Qualitative evaluation of TMFT, CCOT [34], ECO [35], MDNet [3] and VITAL [11] on the OTB sequence, *Bird1*

from TMFT mitigate the drift seen in the other trackers. As a result, TMFT is able to more accurately locate the object with respect to the ground truth centres. The primary challenge in both *Freeman4* and *Bird1* is occlusion. TMFT is able to remember the appearance of the occluded object when the occlusion ends due to its use of the feature history. For example, all the trackers drift by different offsets for the *Freeman4* sequence when the person is occluded. TMFT is the tracker that is able to most accurately relocate the person when he is revealed. For the *Bird1* sequence, TMFT is able to quickly recover and relocate the bird even when everything is obscured by a cloud for a period. Two trackers, CCOT and ECO, fail to track the bird in the *Bird1* sequence and have trouble tracking the person in the *Freeman4* sequence.

These tracking results show the effectiveness of TMFT over the other trackers. Since ECO and CCOT do not attempt to learn how the features change with time, they fail to track objects that change their appearances. VITAL uses an adversarial learning approach to learn appearance changes over time. However, it does not accurately track each object as the proposed TMFT does probably due to not using the full history. TMFT makes use of the history of the target appearance to learn a TMF to capture persistent features, which allow it to better locate the target objects in all sample sequences.

5 | CONCLUSIONS AND FUTURE WORK

We propose a tracker, named TMFT, which organizes the target object's feature history into source and target domains and incorporates domain adaptation into the online learning

process of visual tracking. TMFT allows the use of the positive feature history to learn a TMF, which captures persistent features as the target object's appearance changes over time. The TMF trains the CNN to focus on similar object features over time to improve the model generalization. TMFT is implemented as an end-to-end CNN that learns the TMF when learning the discriminative features of the target object at the same time. Experimental results on the OTB-100 dataset demonstrate that TMFT outperforms 25 state-of-the-art trackers with a centre error precision of 0.928 and outperforms 25 state-of-the-art trackers except for ECO with an overlap success of 0.685. Experimental results on the VOT2016 dataset demonstrate that TMFT outperforms four compared trackers with accuracy of 0.5439, robustness of 10.2333 and EAO of 0.3753. These results also reinforce the findings of [8, 9] to show that online learning with domain adaptation is applicable to the field of visual tracking.

In the future, we will study the effect of the adaptive gradient scale factor on tracking performance and empirically derive an appropriate equation to produce an optimal gradient scale factor for tracking. We will also study different weighting schemes to combine the classification and filter loss functions. We will further investigate better strategies to divide the history into source and target domains and investigate different strategies to improve run-time performance without degrading tracking performance.

ACKNOWLEDGEMENT

This research is supported in part by a Tomorrow Ph.D. Fellowship from Space Dynamics Laboratory.

REFERENCES

1. Wang, N., et al.: Transferring rich feature hierarchies for robust visual tracking arXiv preprint arXiv:150104587 (2015)
2. Hong, S., et al.: Online tracking by learning discriminative saliency map with convolutional neural network. In: Proceedings of the 32nd international conference on machine learning. Vol. 37, pp. 597–606 PMLR, Lille (2015)
3. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4293–4302 (2016)
4. Guo, Q., et al.: Learning dynamic siamese network for visual object tracking. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
5. Javanmardi, M., Qi, X.: Appearance variation adaptation tracker using adversarial network. Neural Netw. 129, 334–343 (2020)
6. Feng, W., et al.: Dynamic saliency-aware regularization for correlation filter-based object tracking. IEEE Trans. Image Process. 28(7), 3232–3245 (2019)
7. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. J. Big Data. 3(1), 9–49 (2016)
8. Cao, Z., et al.: Partial adversarial domain adaptation. In: Computer vision – ECCV, 11212, pp. 139–155. Springer International Publishing, Munich (2018)
9. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Vol. 37 of Proceedings of Machine Learning Research, pp. 1180–1189. Lille, PMLR (2015)
10. Rahman, M.M., et al.: Correlation-aware adversarial domain adaptation and generalization. Pattern Recognit. 100, 107124–107136 (2019)
11. Song, Y., et al.: VITAL: Visual tracking via adversarial learning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8990–8999 (2018)
12. Pu, S., et al.: Deep attentive tracking via reciprocative learning.Advances in Neural Information Processing Systems, 31, pp. 1931–1941. Curran Associates, Inc. (2018)
13. Han, Y., et al.: Robust visual tracking based on adversarial unlabelled instance generation with label smoothing loss regularization. Pattern Recogn. 97, 107027–107051 (2020)
14. Gaidon, A., Vig, E.: Online domain adaptation for multi-object tracking. In: Xie, X., Jones, M.W., Tam, G.K.L. (eds.) Proceedings of the British Machine Vision Conference (BMVC), pp. 31–313. BMVA Press (2015)
15. Jung, I., et al.: Real-time MDNet. In: Computer Vision – ECCV, 11208, pp. 89–104. Springer International Publishing (2018)
16. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: a benchmark. In: 2013 IEEE conference on computer vision and pattern recognition, 2411–2418. (2013)
17. Kristan, M., et al.: A novel performance evaluation methodology for single-target trackers. IEEE Trans. Pattern Anal. Mach. Intell. 38(11), 2137–2155 (2016)
18. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. 115(3), 211–252 (2015)
19. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. IEEE Trans. Pattern Anal. Mach. Intell. 37(9), 1834–1848 (2015)
20. Danelljan, M., et al.: Accurate scale estimation for robust visual tracking. In: Proceedings of the British Machine Vision Conference. pp. 1–11. BMVA Press (2014)
21. Henriques, J.F., et al.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. 37(3), 583–596 (2015)
22. Gao, J., et al.: Transfer learning based visual tracking with Gaussian processes regression. In: Computer Vision – ECCV, 8691, pp. 188–203. Springer (2014)
23. Zhang, J., Ma, S., Sclaroff, S.: MEEM: Robust tracking via multiple experts using entropy minimization. In: Computer Vision – ECCV, vol. 8694, pp. 188–203. Springer (2014)
24. Hong, Z., et al.: MUlti-store tracker (MUSTer): A cognitive psychology inspired approach to object tracking. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 749–758 (2015)
25. Ma, C., et al.: Long-term correlation tracking. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5388–5396 (2015)
26. Zhang, T., Xu, C., Yang, M.H.: Robust structural sparse tracking. IEEE Trans. Pattern Anal. Mach. Intell. 41(2), 473–486 (2019)
27. Danelljan, M., et al.: Learning spatially regularized correlation filters for visual tracking. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 4310–4318 (2015)
28. Bertinetto, L., et al.: Fully-convolutional siamese networks for object tracking. In: Comput. Vision – ECCV 2016 Workshops, 9914, pp. 850–865. Springer (2016)
29. Danelljan, M., et al.: Convolutional features for correlation filter based visual tracking. In: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), pp. 621–629 (2015)
30. Yun, S., et al.: Action-decision networks for visual tracking with deep reinforcement learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1349–1358 (2017)
31. Valmadre, J., et al.: End-to-end representation learning for correlation filter based tracking. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5000–5008 (2017)
32. Javanmardi, M., Qi, X.: Structured group local sparse tracker. IET Image Process. 13(8), 1391–1399 (2019)
33. Choi, J., et al.: Visual tracking using attention-modulated disintegration and integration. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4321–4330 (2016)

34. Danelljan, M., et al.: Beyond correlation filters: learning continuous convolution operators for visual tracking. In: Computer vision – ECCV 2016, pp. 472–488. Springer (2016)
35. Danelljan, M., Bhat, G., KhanFahadShahbazndFelsberg, M.: ECO: Efficient convolution operators for tracking. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6638–6646 (2017)
36. Song, Y., et al.: CREST: Convolutional residual learning for visual tracking. In: 2017 IEEE International Conference on Computer Vision (ICCV), 2574–2583 (2017)
37. Choi, J., et al.: Context-aware deep feature compression for high-speed visual tracking. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 479–488 (2018)
38. Li, B., et al.: High performance visual tracking with siamese region proposal network. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8971–8980 (2018)
39. Bertinetto, L., et al.: Staple: complementary learners for real-time tracking. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1401–1409 (2016)
40. Zhang, K., et al.: Robust visual tracking via convolutional networks without training. IEEE Trans. Image Process. 25(4), 1779–1792 (2016)
41. Qi, Y., et al.: Hedged deep tracking. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4303–4311 (2016)
42. Kristan, M., et al.: The visual object tracking VOT2016 challenge results. In: Computer Vision – ECCV 2016 Workshops. Lecture Notes in Computer Science, pp. 778–823 (2016)
43. Kristan, M., et al.: The sixth visual object tracking VOT2018 challenge results. In: Leal-Taixé, L., Roth, S. (eds.) Computer vision – ECCV 2018 workshops, pp. 3–53.Springer International Publishing, Cham (2019)
44. Kristan, M., et al.: The visual object tracking VOT2015 challenge results. In: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), pp. 564–586 (2015)
45. Xu, T., et al.: Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual object tracking. IEEE Trans. Image Process. 28(11), 5596–5609 (2019)
46. Bai, S., et al.: Multi-hierarchical independent correlation filters for visual tracking. In: 2020 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6 (2020)

How to cite this article: Robeson B, Javanmardi M, Qi X. Object tracking using temporally matching filters. *IET Comput. Vis.* 2021;15:245–257. https://doi.org/10.1049/cvi2.12040