Unsupervised Deep Tracking

Ning Wang[1] Yibing Song[2*] Chao Ma[3] Wengang Zhou[1] Wei Liu[2*] Houqiang Li[1]
[1] CAS Key Laboratory of GIPAS, University of Science and Technology of China
[2] Tencent AI Lab
[3] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
wn6149@mail.ustc.edu.cn, dynamicstevenson@gmail.com, chaoma@sjtu.edu.cn
zhwg@ustc.edu.cn, wl2223@columbia.edu, lihq@ustc.edu.cn

Abstract

*We propose an unsupervised visual tracking method in this paper. Different from existing approaches using extensive annotated data for supervised learning, our CNN model is trained on large-scale unlabeled videos in an unsupervised manner. Our motivation is that a robust tracker should be effective in both the forward and backward predictions (*i.e.*, the tracker can forward localize the target object in successive frames and backtrace to its initial position in the first frame). We build our framework on a Siamese correlation filter network, which is trained using unlabeled raw videos. Meanwhile, we propose a multiple-frame validation method and a cost-sensitive loss to facilitate unsupervised learning. Without bells and whistles, the proposed unsupervised tracker achieves the baseline accuracy of fully supervised trackers, which require complete and accurate labels during training. Furthermore, unsupervised framework exhibits a potential in leveraging unlabeled or weakly labeled data to further improve the tracking accuracy.*

1. Introduction

Visual tracking is a fundamental task in computer vision, which aims to localize the target object in the video given a bounding box annotation in the first frame. The state-of-the-art deep tracking methods [1, 46, 15, 55, 27, 60, 58, 54, 4, 19, 33, 34] typically use pretrained CNN models for feature extraction. These models are trained in a supervised manner, requiring a large quantity of annotated ground-truth labels. Manual annotations are always expensive and time-consuming, whereas extensive unlabeled videos are readily available on the Internet. It deserves to investigate how to exploit unlabeled video sequences for visual tracking.

*Y. Song and W. Liu are the corresponding authors. This work is done when N. Wang is an intern in Tencent AI Lab. The source code and results are available at https://github.com/594422814/UDT.

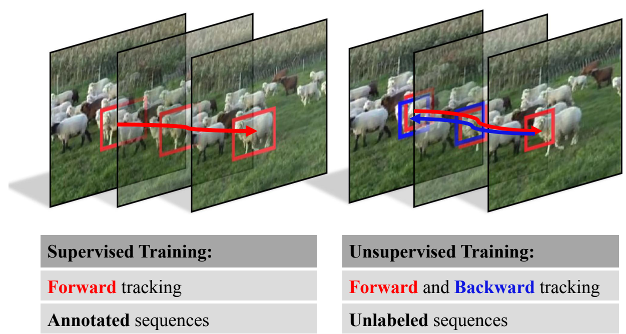

Figure 1. The comparison between supervised and unsupervised learning. Visual tracking methods via supervised learning require ground-truth labels for every frame of the training videos. By utilizing the forward tracking and backward verification, we train the unsupervised tracker without heavyweight annotations.

In this paper, we propose to learn a visual tracking model from scratch via unsupervised learning. Our intuition resides on the observation that visual tracking can be performed in both the forward and backward manners. Initially, given the target object annotated on the first frame, we can track the target object forward in the subsequent frames. When tracking backward, we use the predicted location in the last frame as the initial target annotation and track it backward towards the first frame. The estimated target location in the first frame via backward tracking is expected to be identical with the initial annotation. After measuring the difference between the forward and backward target trajectories, our network is trained in an unsupervised manner[1] by considering the trajectory consistency as shown in Fig. 1. Through exploiting consecutive frames in unlabeled videos, our model learns to locate targets by repeatedly performing forward tracking and backward verification.

The proposed unsupervised learning scheme aims to acquire a generic feature representation, while not being

[1] In this paper, we do not distinguish between the term *unsupervised* and *self-supervised*, as both refer to learning without ground-truth annotations.

strictly required to track a complete object. For a video sequence, we randomly initialize a bounding box in the first frame, which may not cover an entire object. Then, the proposed model learns to track the bounding box region in the following sequences. This tracking strategy shares similarity with the part-based [30] or edge-based [28] tracking methods that focus on tracking the subregions of the target objects. As the visual object tracker is not expected to only concentrate on the complete objects, we use the randomly cropped bounding boxes for tracking initialization during training.

We integrate the proposed unsupervised learning into the Siamese based correlation filter framework [54]. The proposed network consists of two steps in the training process: forward tracking and backward verification. We notice that the backward verification is not always effective since the tracker may successfully return to the initial target location from a deflected or false position. In addition, challenges such as heavy occlusion in unlabeled videos will further degrade the network representation capability. To tackle these issues, we propose multiple frames validation and a cost-sensitive loss to benefit the unsupervised training. The multiple frames validation increases the discrepancy between the forward and backward trajectories to reduce verification failures. Meanwhile, the cost-sensitive loss mitigates the interference from noisy samples during training.

The proposed unsupervised tracker is shown effective on the benchmark datasets. Extensive experimental results indicate that without bells and whistles, the proposed unsupervised tracker achieves comparable performance with the baseline fully supervised trackers [1, 49, 54]. When integrated with additional improvements such as the adaptive online model update [9, 7], the proposed tracker exhibits state-of-the-art performance. It is worth mentioning that the unsupervised framework shows potential in exploiting unlabeled Internet videos to learn good feature representations for tracking scenarios. Given limited or noisy labels, the unsupervised method exhibits comparable results with the corresponding supervised framework. In addition, we further improve the tracking accuracy by using more unlabeled data. Sec. 4.2 shows a complete analysis of different training configurations.

In summary, the contributions of our work are three-fold:

- We propose an unsupervised tracking method based on the Siamese correlation filter backbone, which is learned via forward and backward tracking.
- We propose a multiple-frame validation method and a cost-sensitive loss to improve the unsupervised learning performance.
- The extensive experiments on the standard benchmarks show the favorable performance of the proposed method and reveal the potential of unsupervised learning in visual tracking.

2. Related Work

In this section, we perform a literature review on the deep tracking methods, forward-backward trajectory analysis, and unsupervised representation learning.

Deep Visual Tracking. Existing deep tracking methods either offline learn a specific CNN model for online tracking or simply utilize off-the-shelf deep models (e.g., VGG [43, 3]) for feature extraction. The Siamese trackers [1, 46, 49, 54, 55, 15, 27, 60, 58] formulate the tracking task as a similarity matching process. They typically offline learn a tracking network and do not fine-tune the model online. On the other hand, some trackers adopt off-the-shelf CNN models as the feature extraction backbone. They incrementally train binary classification layers [37, 45, 39] or regression layers [44, 31] based on the initial frame. These methods typically achieve high accuracy while consuming a huge computational cost. The Discriminative Correlation Filter (DCF) based trackers [2, 16, 8, 30, 5, 52, 18] tackle the tracking task by solving a ridge regression problem using densely sampled candidates, which also benefit from the powerful off-the-shelf deep features (e.g., [35, 40, 53, 7]). The main distinction is that deep DCF trackers merely utilize off-the-shelf models for feature extraction and do not online train additional layers or fine-tune the CNN models. Different from the above deep trackers using off-the-shelf models or supervised learning, the proposed method trains a network from scratch using unlabeled data in the wild.

Forward-Backward Analysis. The forward-backward trajectory analysis has been widely explored in the literature. The tracking-learning-detection (TLD) [20] uses the Kanade-Lucas-Tomasi (KLT) tracker [47] to perform forward-backward matching to detect tracking failures. Lee *et al.* [25] proposed to select the reliable base tracker by comparing the geometric similarity, cyclic weight, and appearance consistency between a pair of forward-backward trajectories. However, these methods rely on empirical metrics to identify the target trajectories. In addition, repeatedly performing forward and backward tracking brings in a heavy computational cost for online tracking. Differently, in TrackingNet [36], forward-backward tracking is used for data annotation and tracker evaluation. In this work, we revisit this scheme to train a deep visual tracker in an unsupervised manner.

Unsupervised Representation Learning. Our framework relates to the unsupervised representation learning. In [26], the feature representation is learned by sorting sequences. The multi-layer auto-encoder on large-scale unlabeled data has been explored in [24]. Vondrick *et al.* [50] proposed to anticipate the visual representation of frames in the future. Wang and Gupta [56] used the KCF tracker [16] to pre-process the raw videos, and then selected a pair of tracked images together with another random patch for learning CNNs using a ranking loss. Our method differs from [56] in

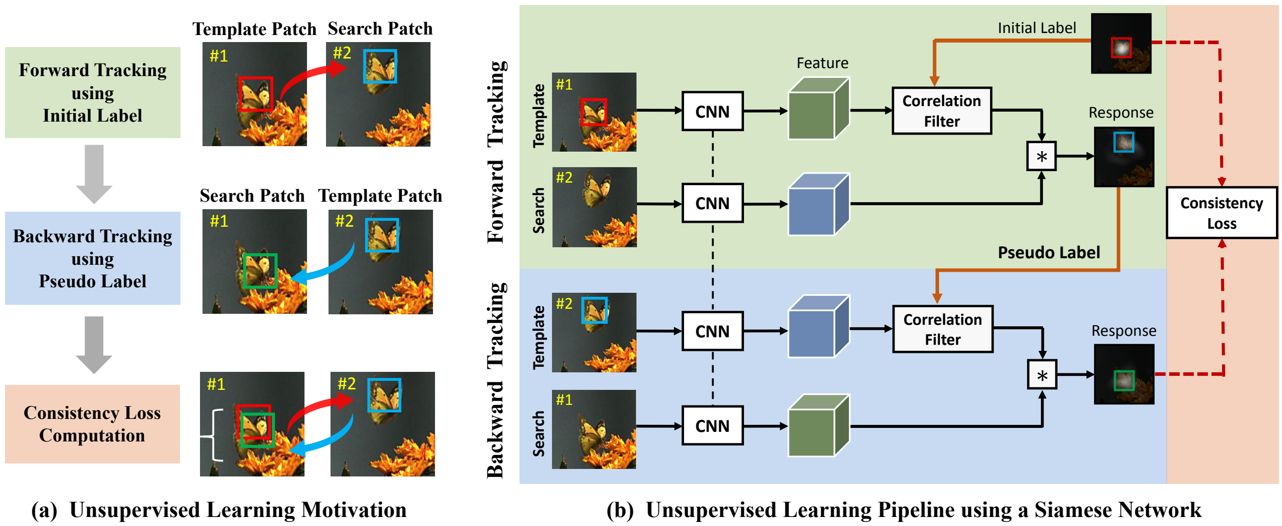

Figure 2. An overview of unsupervised deep tracking. We show our motivation in (a) that we track forward and backward to compute the consistency loss for network training. The detailed training procedure is shown in (b), where unsupervised learning is integrated into a Siamese correlation filter network. Note that during online tracking, we only track forward to predict the target location.

two aspects. First, we integrate the tracking algorithm into unsupervised training instead of merely utilizing an off-the-shelf tracker as the data pre-processing tool. Second, our unsupervised framework is coupled with a tracking objective function, so the learned feature representation is effective in presenting the generic target objects. In the visual tracking community, unsupervised learning has rarely been touched. To the best of our knowledge, the only related but different approach is the auto-encoder based method [51]. However, the encoder-decoder is a general unsupervised framework [38], whereas our unsupervised method is specially designed for tracking tasks.

3. Proposed Method

Fig. 2(a) shows an example from the *Butterfly* sequence to illustrate forward and backward tracking. In practice, we randomly draw bounding boxes in unlabeled videos to perform forward and backward tracking. Given a randomly initialized bounding box label, we first track forward to predict its location in the subsequent frames. Then, we reverse the sequence and take the predicted bounding box in the last frame as the pseudo label to track backward. The predicted bounding box via backward tracking is expected to be identical with the original bounding box in the first frame. We measure the difference between the forward and backward trajectories using the consistency loss for network training. An overview of the proposed unsupervised Siamese correlation filter network is shown in Fig. 2(b). In the following, we first revisit the correlation filter based tracking framework and then illustrate the details of our unsupervised deep tracking approach.

3.1. Revisiting Correlation Tracking

The Discriminative Correlation Filters (DCFs) [2, 16] regress the input features of a search patch to a Gaussian response map for target localization. When training a DCF, we select a template patch $\mathbf{X}$ with the ground-truth label $\mathbf{Y}$. The filter $\mathbf{W}$ can be learned by solving the ridge regression problem as follows:

$$\min_{\mathbf{W}} \|\mathbf{W} * \mathbf{X} - \mathbf{Y}\|_2^2 + \lambda \|\mathbf{W}\|_2^2, \tag{1}$$

where λ is a regularization parameter and $*$ denotes the circular convolution. Eq. 1 can be efficiently calculated in the Fourier domain [2, 8, 16] and the DCF can be computed by

$$\mathbf{W} = \mathscr{F}^{-1}\left(\frac{\mathscr{F}(\mathbf{X}) \odot \mathscr{F}^{\star}(\mathbf{Y})}{\mathscr{F}^{\star}(\mathbf{X}) \odot \mathscr{F}(\mathbf{X}) + \lambda}\right), \tag{2}$$

where $\odot$ is the element-wise product, $\mathscr{F}(\cdot)$ is the Discrete Fourier Transform (DFT), $\mathscr{F}^{-1}(\cdot)$ is the inverse DFT, and $\star$ denotes the complex-conjugate operation. In each subsequent frame, given a search patch $\mathbf{Z}$, the corresponding response map $\mathbf{R}$ can be computed in the Fourier domain:

$$\mathbf{R} = \mathbf{W} * \mathbf{Z} = \mathscr{F}^{-1}\left(\mathscr{F}^{\star}(\mathbf{W}) \odot \mathscr{F}(\mathbf{Z})\right). \tag{3}$$

The above DCF framework starts from learning a target template $\mathbf{W}$ using the template patch $\mathbf{X}$ and then convolves $\mathbf{W}$ with a search patch $\mathbf{Z}$ to generate the response. Recently, the Siamese correlation filter network [49, 54] embeds the DCF in a Siamese framework and constructs two shared-weight branches as shown in Fig. 2(b). The first one is the template branch which takes a template patch $\mathbf{X}$ as input and extracts its features to further generate a target template via DCF. The second one is the search branch which takes a search patch $\mathbf{Z}$ as input for feature extraction. The target template is then convolved with the CNN features of the search patch to generate the response map. The advantage of the Siamese DCF network is that both the feature extraction CNN and correlation filter are formulated into an end-to-end framework, so that the learned features are more related to the visual tracking scenarios.

3.2. Unsupervised Learning Prototype

Given two consecutive frames P_1 and P_2, we crop the template and search patches from them, respectively. By conducting forward tracking and backward verification, the proposed framework does not require ground-truth labeling for supervised training. The difference between the initial bounding box and the predicted bounding box in P_1 will formulate a consistency loss for network learning.

Forward Tracking. We follow [54] to build a Siamese correlation filter network to track the initial bounding box region in frame P_1. After cropping the template patch $\mathbf{T}$ from the first frame P_1, the corresponding target template $\mathbf{W_T}$ can be computed as:

$$\mathbf{W_T} = \mathscr{F}^{-1}\left(\frac{\mathscr{F}(\varphi_\theta(\mathbf{T})) \odot \mathscr{F}^\star(\mathbf{Y_T})}{\mathscr{F}^\star(\varphi_\theta(\mathbf{T})) \odot \mathscr{F}(\varphi_\theta(\mathbf{T})) + \lambda}\right), \quad (4)$$

where $\varphi_\theta(\cdot)$ denotes the CNN feature extraction operation with trainable network parameters θ, and $\mathbf{Y_T}$ is the label of the template patch $\mathbf{T}$. This label is a Gaussian response centered at the initial bounding box center. Once we obtain the learned target template $\mathbf{W_T}$, the response map of a search patch $\mathbf{S}$ from frame P_2 can be computed by

$$\mathbf{R_S} = \mathscr{F}^{-1}(\mathscr{F}^\star(\mathbf{W_T}) \odot \mathscr{F}(\varphi_\theta(\mathbf{S}))). \quad (5)$$

If the ground-truth Gaussian label of patch $\mathbf{S}$ is available, the network $\varphi_\theta(\cdot)$ can be trained by computing the L_2 distance between $\mathbf{R_S}$ and the ground-truth. In the following, we show how to train the network without labels by exploiting backward trajectory verification.

Backward Tracking. After generating the response map $\mathbf{R_S}$ for frame P_2, we create a pseudo Gaussian label centered at its maximum value, which is denoted by $\mathbf{Y_S}$. In backward tracking, we switch the role between the search patch and the template patch. By treating $\mathbf{S}$ as the template patch, we generate a target template $\mathbf{W_S}$ using the pseudo label $\mathbf{Y_S}$. The target template $\mathbf{W_S}$ can be learned using Eq. (4) by replacing $\mathbf{T}$ with $\mathbf{S}$ and replacing $\mathbf{Y_T}$ with $\mathbf{Y_S}$. Then, we generate the response map $\mathbf{R_T}$ through Eq. (5) by replacing $\mathbf{W_T}$ with $\mathbf{W_S}$ and replacing $\mathbf{S}$ with $\mathbf{T}$. Note that we only use one Siamese correlation filter network to track forward and backward. The network parameters θ are fixed during the tracking steps.

Consistency Loss Computation. After forward and backward tracking, we obtain the response map $\mathbf{R_T}$. Ideally, $\mathbf{R_T}$ should be a Gaussian label with the peak located at the initial target position. In other words, $\mathbf{R_T}$ should be as similar as the originally given label $\mathbf{Y_T}$. Therefore, the representation network $\varphi_\theta(\cdot)$ can be trained in an unsupervised manner by minimizing the reconstruction error as follows:

$$\mathcal{L}_{\text{un}} = \|\mathbf{R_T} - \mathbf{Y_T}\|_2^2. \quad (6)$$

We perform back-propagation of the computed loss to update the network parameters. During back-propagation, we follow the Siamese correlation filter methods [54, 59] to update the network as:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{un}}}{\partial \varphi_\theta(\mathbf{T})} &= \mathscr{F}^{-1}\left(\frac{\partial \mathcal{L}_{\text{un}}}{\partial\left(\mathscr{F}\left(\varphi_\theta(\mathbf{T})\right)\right)^\star} + \left(\frac{\partial \mathcal{L}_{\text{un}}}{\partial\left(\mathscr{F}\left(\varphi_\theta(\mathbf{T})\right)\right)}\right)^\star\right), \\ \frac{\partial \mathcal{L}_{\text{un}}}{\partial \varphi_\theta(\mathbf{S})} &= \mathscr{F}^{-1}\left(\frac{\partial \mathcal{L}_{\text{un}}}{\partial\left(\mathscr{F}\left(\varphi_\theta(\mathbf{S})\right)\right)^\star}\right).\end{aligned} \quad (7)$$

3.3. Unsupervised Learning Improvements

The proposed unsupervised learning method constructs the objective function based on the consistency between $\mathbf{R_T}$ and $\mathbf{Y_T}$. In practice, the tracker may deviate from the target in the forward tracking but still return to the original position during the backward process. However, the proposed loss function does not penalize this deviation because of the consistent predictions. Meanwhile, the raw videos may contain uninformative or even corrupted training samples with occlusion that deteriorate the unsupervised learning process. We propose multiple frames validation and a cost-sensitive loss to tackle these limitations.

3.3.1 Multiple Frames Validation

We propose a multiple frames validation approach to alleviate the inaccurate localization issue that is not penalized by Eq. (6). Our intuition is to involve more frames during forward and backward tracking to reduce the verification failures. The reconstruction error in Eq. (6) tends to be amplified and the computed loss will facilitate the training process.

During unsupervised learning, we involve another frame P_3 which is the subsequent frame after P_2. We crop a search patch $\mathbf{S}_1$ from P_2 and another search patch $\mathbf{S}_2$ from P_3. If the generated response map $\mathbf{R}_{\mathbf{S}_1}$ is different from its corresponding ground-truth response, this error tends to become larger in the next frame P_3. As a result, the consistency is more likely to be broken in the backward tracking, and the generated response map $\mathbf{R_T}$ is more likely to deviate from $\mathbf{Y_T}$. By simply involving more search patches during forward and backward tracking, the proposed consistency loss

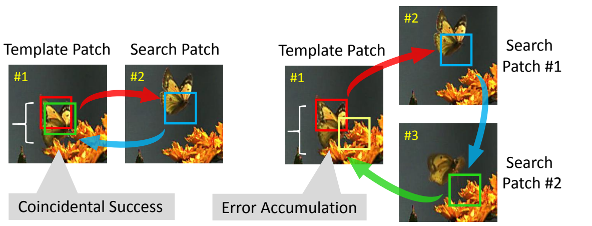

Figure 3. Single frame validation and multiple frames validation. The inaccurate localization in single frame validation may not be captured as shown on the left. By involving more frames as shown on the right, we can accumulate the localization error to break the prediction consistency during forward and backward tracking.

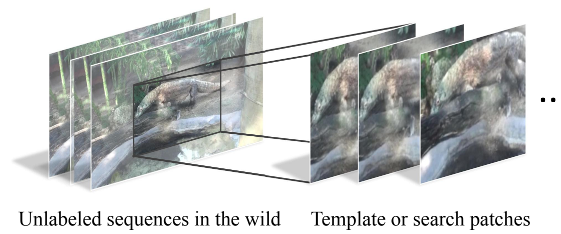

Figure 4. An illustration of training samples generation. The proposed method simply crops and resizes the center regions from unlabeled videos as the training patches.

will be more effective to penalize the inaccurate localizations as shown in Fig. 3. In practice, we use three frames to validate and the improved consistency loss is written as:

$$\mathcal{L}_{\text{un}} = \|\widetilde{\mathbf{R}}_{\mathbf{T}} - \mathbf{Y}_{\mathbf{T}}\|_2^2, \tag{8}$$

where $\widetilde{\mathbf{R}}_{\mathbf{T}}$ is the response map generated by an additional frame during the backward tracking step.

3.3.2 Cost-sensitive Loss

We randomly initialize a bounding box region in the first frame P_1 for forward tracking. This bounding box region may contain noisy background context (e.g., occluded targets). Fig. 5 shows an overview of these regions. To alleviate the background interference, we propose a cost-sensitive loss to exclude noisy samples for network training.

During unsupervised learning, we construct multiple training pairs from the training sequences. Each training pair consists of one initial template patch $\mathbf{T}$ in frame P_1 and two search patches $\mathbf{S}_1$ and $\mathbf{S}_2$ from the subsequent frames P_2 and P_3, respectively. These training pairs form a training batch to train the Siamese network. In practice, we find that few training pairs with extremely high losses prevent the network training from convergence. To reduce the contributions of noisy pairs, we exclude 10% of the whole training pairs which contain a high loss value. Their losses can be computed using Eq. (8). To this end, we assign a binary weight $\mathbf{A}^i_{\text{drop}}$ to each training pair and all the weight elements form the weight vector $\mathbf{A}_{\text{drop}}$. The 10% of its elements are 0 and the others are 1.

In addition to the noisy training pairs, the raw videos include lots of uninformative image patches which only contain the background or still targets. For these patches, the objects (e.g., sky, grass, or tree) hardly move. Intuitively, the target with a large motion contributes more to the network training. Therefore, we assign a motion weight vector $\mathbf{A}_{\text{motion}}$ to all the training pairs. Each element $\mathbf{A}^i_{\text{motion}}$ can be computed by

$$\mathbf{A}^i_{\text{motion}} = \left\|\mathbf{R}^i_{\mathbf{S}_1} - \mathbf{Y}^i_{\mathbf{T}}\right\|_2^2 + \left\|\mathbf{R}^i_{\mathbf{S}_2} - \mathbf{Y}^i_{\mathbf{S}_1}\right\|_2^2, \tag{9}$$

where $\mathbf{R}^i_{\mathbf{S}_1}$ and $\mathbf{R}^i_{\mathbf{S}_2}$ are the response maps in the i-th training pair, $\mathbf{Y}^i_{\mathbf{T}}$ and $\mathbf{Y}^i_{\mathbf{S}_1}$ are the corresponding initial and pseudo labels, respectively. Eq. (9) calculates the target motion difference from frame P_1 to P_2 and P_2 to P_3. The larger value of $\mathbf{A}^i_{\text{motion}}$ indicates that the target undergoes a larger movement in this continuous trajectory. On the other hand, we can interpret that the large value of $\mathbf{A}^i_{\text{motion}}$ represents the hard training pair which the network should pay more attentions to. We normalize the motion weight and the binary weight as follows,

$$\mathbf{A}^i_{\text{norm}} = \frac{\mathbf{A}^i_{\text{drop}} \cdot \mathbf{A}^i_{\text{motion}}}{\sum_{i=1}^{n} \mathbf{A}^i_{\text{drop}} \cdot \mathbf{A}^i_{\text{motion}}}, \tag{10}$$

where n is number of the training pairs in a mini-batch. The final unsupervised loss in a mini-batch is computed as:

$$\mathcal{L}_{\text{un}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{A}^i_{\text{norm}} \cdot \left\|\widetilde{\mathbf{R}}^i_{\mathbf{T}} - \mathbf{Y}^i_{\mathbf{T}}\right\|_2^2. \tag{11}$$

3.4. Unsupervised Training Details

Network Structure. We follow the DCFNet [54] to use a shallow Siamese network with only two convolutional layers. The filter sizes of these convolutional layers are $3 \times 3 \times 3 \times 32$ and $3 \times 3 \times 32 \times 32$, respectively. Besides, a local response normalization (LRN) layer is employed at the end of convolutional layers. This lightweight structure enables extremely efficient online tracking.

Training Data. We choose the widely used ILSVRC 2015 [42] as our training data to fairly compare with existing supervised trackers. In the data pre-processing step, existing supervised approaches [1, 49, 54] require ground-truth labels for every frame. Meanwhile, they usually discard the frames where the target is occluded, or the target is partially out of view, or the target infrequently appears in tracking scenarios (e.g., snake). This requires a time-consuming human interaction to preprocess the training data.

In contrast, we do not preprocess any data and simply crop the center patch in each frame. The patch size is the half of the whole image and further resized to 125×125 as

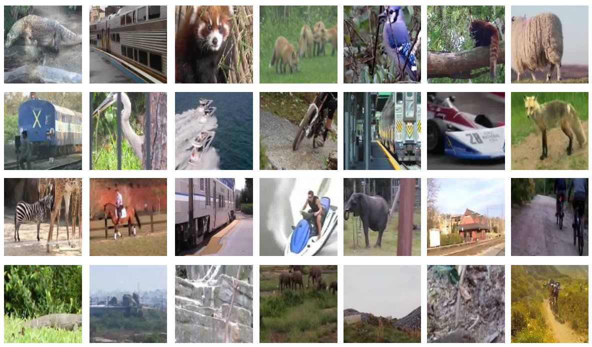

Figure 5. Examples of randomly cropped center patches from ILSVRC 2015 [42]. Most patches contain valuable contents while some are less meaningful (e.g., the patches on the last row).

the network input as shown in Fig. 4. We randomly choose three cropped patches from the continuous 10 frames in a video. We set one of the three patches as the template and the remaining as search patches. This is based on the assumption that the center located target objects are unlikely to move out of the cropped region in a short period. We track the objects appearing in the center of the cropped regions, while not specifying their categories. Some examples of the cropped regions are exhibited in Fig. 5.

3.5. Online Object Tracking

After offline unsupervised learning, we online track the target object following forward tracking as illustrated in Sec. 3.2. To adapt the object appearance variations, we online update the DCF parameters as follows:

$$\mathbf{W}_t = (1 - \alpha_t)\mathbf{W}_{t-1} + \alpha_t \mathbf{W}, \tag{12}$$

where $\alpha_t \in [0, 1]$ is the linear interpolation coefficient. The target scale is estimated through a patch pyramid with scale factors $\{a^s | a = 1.015, s = \{-1, 0, 1\}\}$ following [10]. We denote the proposed Unsupervised Deep Tracker as UDT, which merely uses standard incremental model update and scale estimation. Furthermore, we use an advanced model update that adaptively changes α_t as well as a better DCF formulation following [7]. The improved tracker is denoted as UDT+.

4. Experiments

In this section, we first analyze the effectiveness of our unsupervised learning framework. Then, we compare with state-of-the-art trackers on the standard benchmarks including OTB-2015 [57], Temple-Color [29] and VOT-2016 [21].

4.1. Experimental Details

In our experiments, we use the stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 0.005 to train our model. Our unsupervised network is trained for 50 epoches with a learning rate exponentially decays from 10^{-2} to 10^{-5} and a mini-batch size of 32. All the experiments are executed on a computer with 4.00GHz Intel Core I7-4790K and NVIDIA GTX 1080Ti GPU.

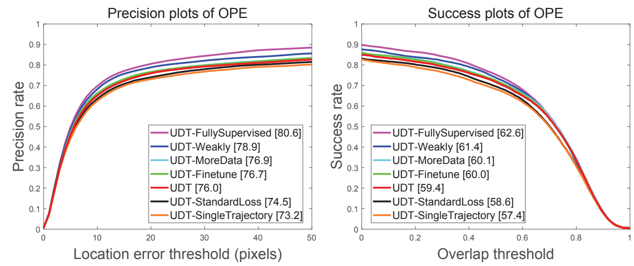

Figure 6. The precision and success plots of our UDT tracker with different configurations on the OTB-2015 dataset [57]. In the legend, we show the distance precision at 20 pixels threshold and area-under-curve (AUC) score.

On the OTB-2015 [57] and TempleColor [29] datasets, we use one-pass evaluation (OPE) with distance precision (DP) at 20 pixels and the area-under-curve (AUC) of the overlap success plot. On the VOT2016 [21], we measure the performance using the Expected Average Overlap (EAO).

4.2. Ablation Study and Analysis

Unsupervised and supervised learning. We use the same training data [42] to train our network via fully supervised learning. Fig. 6 shows the evaluation results where the fully supervised training configuration improves UDT by 3% under the AUC scores.

Stable training. We analyze the effectiveness of our stable training by using different configurations. Fig. 6 shows the evaluation results of multiple learned trackers. The UDT-StandardLoss indicates the results from the tracker learned without using hard sample reweighing (i.e., $\mathbf{A}_{\text{motion}}$ in Eq. (9)). The UDT-SingleTrajectory denotes the results from the tracker learned only using the prototype framework in Sec. 3.2. The results show that multiple frames validation and cost-sensitive loss improve the accuracy.

Using high-quality training data. We analyze the performance variations by using high-quality training data. In ILSVRC 2015 [42], instead of randomly cropping patches, we add offsets ranging from [-20, +20] pixels to the ground-truth bounding boxes for training samples collection. These patches contain more meaningful objects than the randomly cropped ones. The results in Fig. 6 show that our tracker learned using weakly labeled samples (i.e., UDT-Weakly) produce comparable results with the supervised configuration. Note that the predicted target location by existing object detectors or optical flow estimators is normally within 20 pixels offset with respect to the ground-truth. These results indicate that UDT achieves comparable performance with supervised configuration when using less ac-

Table 1. Comparison results with fully-supervised baseline (left) and state-of-the-art (right) trackers on the OTB-2015 benchmark [57]. The evaluation metric is AUC score. Our unsupervised UDT tracker performs favorably against baseline methods shown on the left, while our UDT+ tracker achieves comparable results with the recent state-of-the-art supervised trackers shown on the right.

Trackers	SiamFC [1]	DCFNet [54]	CFNet [49]	**UDT**	DSiam [14]	EAST [17]	HP [13]	SA-Siam [15]	SiamPRN [27]	RASNet [55]	SACF [59]	Siam-tri [12]	RT-MDNet [19]	MemTrack [58]	StructSiam [60]	**UDT+**
AUC score (%)	58.2	58.0	56.8	59.4	60.5	62.9	60.1	65.7	63.7	64.2	63.3	59.2	65.0	62.6	62.1	63.2
Speed (FPS)	86	70	65	70	25	159	69	50	160	83	23	86	50	50	45	55

curate labels produced by existing detection or flow estimation methods.

Few-shot domain adaptation. We collect the first 5 frames from the videos in OTB-2015 [57] with only the ground-truth bounding box available in the first frame. Using these limited samples, we fine-tune our network by 100 iterations using the forward-backward pipeline. This training process takes around 6 minutes. The results (i.e., UDT-Finetune) show that the performance is further enhanced. Our offline unsupervised training learns general feature representation, which can be transferred to a specific domain (e.g., OTB) using few-shot adaptation. This domain adaptation is similar to MDNet [37] but our initial parameters are offline learned in an unsupervised manner.

Adopting more unlabeled data. Finally, we utilize more unlabeled videos for network training. These additional raw videos are from the OxUvA benchmark [48] (337 videos in total), which is a subset of Youtube-BB [41]. In Fig. 6, our UDT-MoreData tracker gains performance improvement (0.9% DP and 0.7% AUC), which illustrates unlabeled data can advance the unsupervised training. Nevertheless, in the following we remain using the UDT and UDT+ trackers which are only trained on [42] for fair comparisons.

4.3. State-of-the-art Comparison

OTB-2015 Dataset. We evaluate the proposed UDT and UDT+ trackers with state-of-the-art real-time trackers including ACT [4], ACFN [6], CFNet [49], SiamFC [1], SCT [5], CSR-DCF [32], DSST [8], and KCF [16] using precision and success plots metrics. Fig. 7 and Table 1 show that the proposed unsupervised tracker UDT is comparable with the baseline supervised methods (i.e., SiamFC and CFNet). Meanwhile, the proposed UDT tracker exceeds DSST algorithm by a large margin. As DSST is a DCF based tracker with accurate scale estimation, the performance improvement indicates that our unsupervised feature representation is more effective than empirical features. In Fig. 7 and Table 1, we do not compare with some remarkable non-realtime trackers. For example, MDNet [37] and ECO [7] can yield 67.8% and 69.4% AUC on the OTB-2015 dataset, but they are far from real-time.

In Table 1, we also compare with more recently proposed supervised trackers. These latest approaches are mainly based on the Siamese network and trained using ILSVRC [42]. Some trackers (e.g., SA-Siam [15] and RT-MDNet [19]) adopt pre-trained CNN models (e.g., AlexNet [23] and VGG-M [3]) for network initialization. The SiamRPN [27] additionally uses more labeled training videos from Youtube-BB dataset [41]. Compared with existing methods, the proposed UDT+ tracker does not require data labels or off-the-shelf deep models while still achieving comparable performance and efficiency.

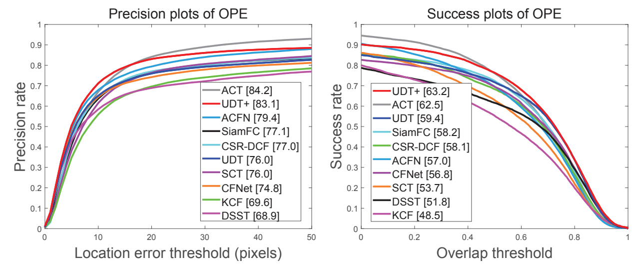

Figure 7. Precision and success plots on the OTB-2015 dataset [57] for recent real-time trackers.

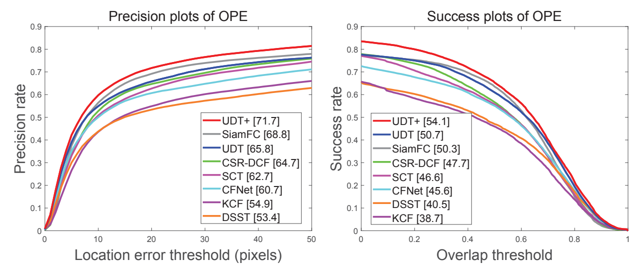

Figure 8. Precision and success plots on the Temple-Color dataset [29] for recent real-time trackers.

Temple-Color Dataset. The Temple-Color [29] is a more challenging benchmark with 128 color videos. We compare our method with the state-of-the-art trackers illustrated in Sec. 4.3. The propose UDT tracker performs favorably against SiamFC and CFNet as shown in Fig. 8.

VOT2016 Dataset. Furthermore, we report the evaluation results on the VOT2016 benchmark [21]. The expected average overlap (EAO) is the final metric for tracker ranking according to the VOT report [22]. As shown in Table 2, the performance of our UDT tracker is comparable with the baseline trackers (e.g., SiamFC). The improved UDT+ tracker performs favorably against state-of-the-art fully-supervised trackers including SA-Siam [15], StructSiam [60] and MemTrack [58].

Attribute Analysis. On the OTB-2015 benchmark, we fur-

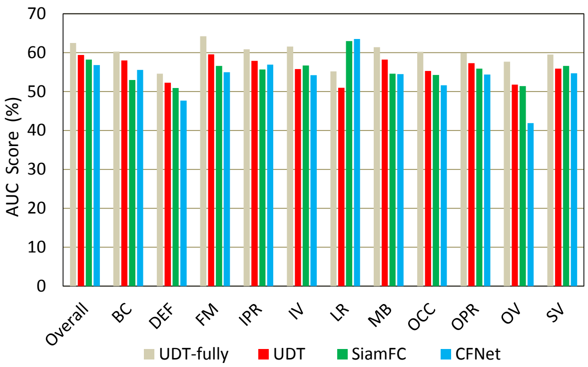

Figure 9. Attribute-based evaluation on the OTB-2015 dataset [57]. The 11 attributes are background clutter (BC), deformation (DEF), fast motion (FM), in-plane rotation (IPR), illumination varition (IV), low resolution (LR), motion blur (MB), occlusion (OCC), out-of-plane rotation (OPR), out-of-view (OV), and scale varition (SV), respectively.

Table 2. Comparison with state-of-the-art and baseline trackers on the VOT2016 benchmark [21]. The evaluation metrics include Accuracy, Failures (over 60 sequences), and Expected Average Overlap (EAO). The up arrows indicate that higher values are better for the corresponding metric and vice versa.

Trackers	Accuracy (↑)	Failures (↓)	EAO (↑)	FPS (↑)
ECO [7]	0.54	-	0.374	6
C-COT [11]	0.52	51	0.331	0.3
pyMDNet [37]	-	-	0.304	2
SA-Siam [15]	0.53	-	0.291	50
StructSiam [60]	-	-	0.264	45
MemTrack [58]	0.53	-	0.273	50
SiamFC [1]	0.53	99	0.235	86
SCT [5]	0.48	117	0.188	40
DSST [8]	0.53	151	0.181	25
KCF [16]	0.49	122	0.192	170
UDT (Ours)	0.54	102	0.226	70
UDT+ (Ours)	0.53	66	0.301	55

ther analyze the performance variations over different challenges as shown in Fig. 9. On the majority of challenging scenarios, the proposed UDT tracker outperforms the SiamFC and CFNet trackers. Compared with the fully-supervised UDT tracker, the unsupervised UDT does not achieve similar tracking accuracies under illumination variation (IV), occlusion (OCC), and fast motion (FM) scenarios. This is because the target appearance variations are significant in these video sequences. Without strong supervision, the proposed tracker is not effective to learn a robust feature representation to overcome these variations.

Qualitative Evaluation. We visually compare the proposed UDT tracker to some supervised trackers (e.g., ACFN, SiamFC, and CFNet) and a baseline DCF tracker (DSST) on eight challenging video sequences. Although the proposed UDT tracker does not employ online improvements, we still observe that UDT effectively tracks the target, especially

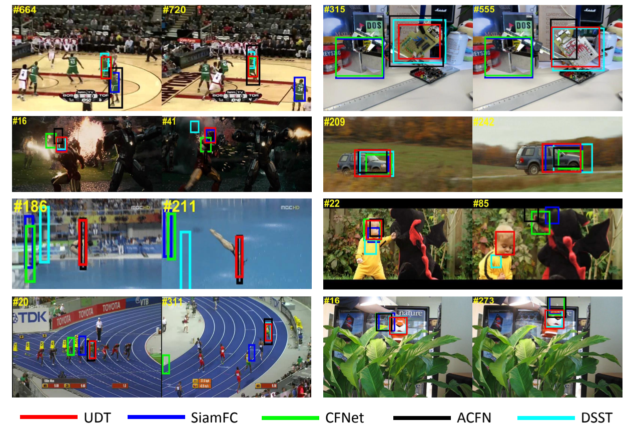

Figure 10. Qualitative evaluation of our proposed UDT and other trackers including SiamFC [1], CFNet [49], ACFN [6], and DSST [8] on 8 challenging videos from OTB-2015. From left to right and top to down are *Basketball*, *Board*, *Ironman*, *CarScale*, *Diving*, *DragonBaby*, *Bolt*, and *Tiger1*, respectively.

on the challenging *Ironman* and *Diving* video sequences as shown in Fig. 10. It is worth mentioning that such a robust tracker is learned using unlabeled videos without ground-truth supervisions.

Limitation. (1) As discussed in the Attribute Analysis, our unsupervised feature representation may lack the objectness information to cope with complex scenarios. (2) Since our approach involves both forward and backward tracking, the computational load is another potential drawback.

5. Conclusion

In this paper, we proposed how to train a visual tracker using unlabeled video sequences in the wild, which has rarely been investigated in visual tracking. By designing an unsupervised Siamese correlation filter network, we verified the feasibility and effectiveness of our forward-backward based unsupervised training pipeline. To further facilitate the unsupervised training, we extended our framework to consider multiple frames and employ a cost-sensitive loss. Extensive experiments exhibit that the proposed unsupervised tracker, without bells and whistles, performs as a solid baseline and achieves comparable results with the classic fully-supervised trackers. Finally, unsupervised framework shows attractive potentials in visual tracking, such as utilizing more unlabeled data or weakly labeled data to further improve the tracking accuracy.

Acknowledgements. This work was supported in part to Dr. Houqiang Li by the 973 Program under Contract No. 2015CB351803 and NSFC under contract No. 61836011, and in part to Dr. Wengang Zhou by NSFC under contract No. 61822208 and 61632019, Young Elite Scientists Sponsorship Program By CAST (2016QNRC001), and the Fundamental Research Funds for the Central Universities. This work was supported in part by National Key Research and Development Program of China (2016YFB1001003), STCSM(18DZ1112300).

References

[1] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

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

[3] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[4] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu. Real-time'actor-critic'tracking. In *ECCV*, 2018.

[5] Jongwon Choi, Hyung Jin Chang, Jiyeoup Jeong, Yiannis Demiris, and Jin Young Choi. Visual tracking using attention-modulated disintegration and integration. In *CVPR*, 2016.

[6] Jongwon Choi, Hyung Jin Chang, Sangdoo Yun, Tobias Fischer, Yiannis Demiris, and Jin Young Choi. Attentional correlation filter network for adaptive visual tracking. In *CVPR*, 2017.

[7] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *CVPR*, 2017.

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

[9] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking. In *CVPR*, 2016.

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

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

[12] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *ECCV*, 2018.

[13] Xingping Dong, Jianbing Shen, Wenguan Wang, Yu Liu, Ling Shao, and Fatih Porikli. Hyperparameter optimization for tracking with continuous deep q-learning. In *CVPR*, 2018.

[14] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In *ICCV*, 2017.

[15] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *CVPR*, 2018.

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

[17] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, 2017.

[18] Jianglei Huang and Wengang Zhou. Re2ema: Regularized and reinitialized exponential moving average for target model update in object tracking. In *AAAI*, 2019.

[19] Ilchae Jung, Jeany Son, Mooyeol Baek, and Bohyung Han. Real-time mdnet. In *ECCV*, 2018.

[20] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *TPAMI*, 34(7):1409–1422, 2012.

[21] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernández, Tomas Vojir, Hager, and et al. The visual object tracking vot2016 challenge results. In *ECCV Workshop*, 2016.

[22] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomáš Vojíř, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *TPAMI*, 38(11):2137–2155, 2016.

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

[24] Quoc V Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, 2013.

[25] Dae Youn Lee, Jae Young Sim, and Chang Su Kim. Multihypothesis trajectory analysis for robust visual tracking. In *CVPR*, 2015.

[26] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017.

[27] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *CVPR*, 2018.

[28] Feng Li, Yingjie Yao, Peihua Li, David Zhang, Wangmeng Zuo, and Ming-Hsuan Yang. Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In *ICCV Workshop*, 2017.

[29] Pengpeng Liang, Erik Blasch, and Haibin Ling. Encoding color information for visual tracking: algorithms and benchmark. *TIP*, 24(12):5630–5644, 2015.

[30] Si Liu, Tianzhu Zhang, Xiaochun Cao, and Changsheng Xu. Structural correlation filter for robust visual tracking. In *CVPR*, 2016.

[31] Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. In *ECCV*, 2018.

[32] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *CVPR*, 2017.

[33] Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *TPAMI*, 2019.

[34] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618*, 2014.

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

[36] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *ECCV*, 2018.

[37] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[38] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[39] Shi Pu, Yibing Song, Chao Ma, Honggang Zhang, and Ming-Hsuan Yang. Deep attentive tracking via reciprocative learning. In *NeurIPS*, 2018.

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

[41] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[44] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *ICCV*, 2017.

[45] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson W.H. Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *CVPR*, 2018.

[46] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[47] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. 1991.

[48] Jack Valmadre, Luca Bertinetto, João F Henriques, Ran Tao, Andrea Vedaldi, Arnold Smeulders, Philip Torr, and Efstratios Gavves. Long-term tracking in the wild: A benchmark. In *ECCV*, 2018.

[49] Jack Valmadre, Luca Bertinetto, João F Henriques, Andrea Vedaldi, and Philip HS Torr. End-to-end representation learning for correlation filter based tracking. In *CVPR*, 2017.

[50] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016.

[51] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, 2013.

[52] Ning Wang, Wengang Zhou, and Houqiang Li. Reliable redetection for long-term tracking. *TCSVT*, 2019.

[53] Ning Wang, Wengang Zhou, Qi Tian, Richang Hong, Meng Wang, and Houqiang Li. Multi-cue correlation filters for robust visual tracking. In *CVPR*, 2018.

[54] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu. Dcfnet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*, 2017.

[55] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. Learning attentions: Residual attentional siamese network for high performance online visual tracking. In *CVPR*, 2018.

[56] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

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

[58] Tianyu Yang and Antoni B Chan. Learning dynamic memory networks for object tracking. In *ECCV*, 2018.

[59] Mengdan Zhang, Qiang Wang, Junliang Xing, Jin Gao, Peixi Peng, Weiming Hu, and Steve Maybank. Visual tracking via spatially aligned correlation filters network. In *ECCV*, 2018.

[60] Yunhua Zhang, Lijun Wang, Jinqing Qi, Dong Wang, Mengyang Feng, and Huchuan Lu. Structured siamese network for real-time visual tracking. In *ECCV*, 2018.