24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Maximum Entropy Reinforced Single Object Visual Tracking

Chenghuan Liu and **Du Q. Huynh** and **Mark Reynolds**
The University of Western Australia
chenghuan.liu@research.uwa.edu.au, {du.huynh, mark.reynolds}@uwa.edu.au

Abstract. Single object visual tracking is a fundamental problem in computer vision and has many applications. Given only the location of the target of interest in the first video frame, a visual tracking algorithm must track the target until the end of the video while having to face challenging factors such as illumination change and scale variation. In this paper, we formulate this tracking problem in a framework of maximum entropy reinforcement learning where the agent is our visual tracker and the goal is to learn a tracking policy that maximises both the expected reward and its entropy so as to achieve a balance between exploitation and exploration. The aim of our tracking framework is to improve the tracking accuracy while giving the tracking agent the ability to avoid getting stuck on a non-target object. Extensive experiments have been performed on a range of benchmarks where our method achieves state-of-the-art performance. Furthermore, we demonstrate that, in contrast to other visual trackers based on deep reinforcement learning, our method can run in real-time while maintaining high tracking accuracy.

1 Introduction

Given the initial state of a target, usually its size and position in the first frame of the video, the goal of visual tracking is to estimate the position and size of the target in each of the following frames. Visual Tracking has been a fundamental task in computer vision and used in various applications. Even though various public benchmarks [40, 17, 17, 27, 10] show that significant progress on visual tracking has been made in the last decades, the performances of many tracking algorithms are still affected by challenging factors such as illumination change, deformation, and scale variation.

Recently, response maps obtained from, for instance, correlation filtering, have been used in many tracking algorithms [4, 8, 15]. The response maps help to simplify the tracking process as the translated location of the target is normally found at the pixel whose response value is the highest. Ideally, only one local maximum is present in the response map for each video frame so that the visual tracker can distinguish the target from the background perfectly. In reality, the response map is more likely to be very noisy and have several local maxima due to various reasons such as similar objects in the background or deformed appearance of the target. As a result, the visual tracker may get stuck in the nearest local maximum that could be a non-target object and is not able to get back to the correct target. Intuitively, to avoid falling onto and getting stuck on a wrong local maximum, the tracker should enlarge its search area to find the target when it is lost (if it is not occluded and its appearance has not changed). However, this contradicts the core task of a visual tracker which is to accurately estimate the state of the target. As the search area is expanded, more distractions would be introduced, making the decision-making process even harder for the visual tracker.

Visual trackers based on reinforcement learning (RL) have been explored to improve the tracking performance against these challenges. ADNet [41] searches for the target in each frame through actions repetitively sampled from the trained network. This requires multiple references to the network and is not efficient. Chen et al. [5] proposed ACT which is trained within the traditional actor-critic reinforcement learning framework. While ACT aims at getting good accuracy (exploitation) only, the problem of getting stuck on the non-target objects (exploration) is not well addressed. To fill these research gaps, we formulate visual tracking in a maximum entropy reinforcement learning framework. The sequential decision-making process of reporting the target's position and size is a typical Markov Decision Process (MDP), where our tracker is the agent that keeps interacting with the environment with actions sampled from the learned policy. In the learning phase, the agent explores the state and action spaces to find the optimal policy, penalising on losing the tracked target or tracking the non-target objects (exploration), while aiming at maintaining high tracking accuracy (exploitation). The deep neural network that is trained for the optimal policy allows the balance between exploration and exploitation to be achieved. The framework is thus able to yield tracking results that have the highest expected reward as well as ensure that the entropy, i.e., the randomness of the policy, is maximised. With "maximum entropy" being incorporated, our method is demonstrated to be more effective than previous RL-based visual trackers with its competitive performances on public benchmarks. Furthermore, different from the ADNet method that tracks the target by making repetitive actions in each video frame, our tracker only needs to perform one action in each frame (both in training and testing), making it more efficient for online tracking.

Our contributions can be summarised as follows:

- In our method, the visual tracking problem is formulated in a maximum entropy reinforcement learning framework. The idea of taking "maximum entropy" into consideration is demonstrated in our experiments in that our method is more effective than previous RL-based visual trackers.
- We design in our RL-based visual tracker a novel representation of the "state", which captures both the information of the target's appearance as well as the response of its interaction with the background.
- We learn the tracking policy from the response map generated by a correlation filter to help the visual tracker avoid getting stuck on a non-target object during tracking.

In the rest of the paper, we briefly review papers that are related to the proposed method and then describe how our visual tracking process is integrated into the maximum entropy reinforcement learning framework. We evaluate our method on five popular benchmarks against state-of-the-art algorithms to demonstrate its competitive performance.

2 Related Work

2.1 Visual Tracking

Currently, there are mainly two kinds of visual trackers: trackers based on the correlation filter and trackers based on the siamese network. The seminal work of visual trackers based on correlation filter is the MOSSE filter [4] which has been extended by many methods, such as introducing kernel trick [15], handling scale changes [9], periodic effect alleviation [7], learning a transformation matrix for model update [16], and using a 4-DoF similarity transformation [21]. Instead of the greyscale features used in MOSSE, HOG [15], fused features [2], and deep features [8] have also been incorporated into the correlation filter.

The siamese trackers have received increasing attention recently thanks to their good balance between speed and accuracy. In these trackers, a two-branch network takes both the candidate and target templates as inputs to generate a score map where the highest value most likely corresponds to the target's position. Attribute-based CNN [30], fully connected network [3], correlation filter [38, 36] and Region Proposal Network (RPN) [19] are also introduced into the siamese tracking framework.

2.2 Reinforcement Learning

The goal of reinforcement learning (RL) is to obtain a policy to maximise the expected rewards by taking sequential actions to interact with the environment [34]. Inspired by the recent successes in deep learning, reinforcement learning has been able to solve many difficult problems such as Atari games [25, 26] as well as a range of computer vision applications like object detection [24], action recognition [35], and person re-identification [42].

There are mainly two families of approaches in reinforcement learning: value-based methods [25] and policy-based methods [1]. Recently, actor-critic methods, which trade off the reduction of the variance of policy gradients in policy-based methods with the bias introduced from value-based methods, have been proposed, e.g., the DPG method [32] explores the deterministic policies instead of stochastic policies normally used in reinforcement learning with continuous action space. Following that, the DDPG method [22] combines DPG with the deep Q-Network (DQN) [25, 26] to concurrently learn a Q-function and a policy. The soft-actor-critic (SAC) method [13] later adopts a stochastic actor in an actor-critic framework to trade-off between exploration and exploitation.

2.3 Visual Tracking Based on Reinforcement Learning

Several attempts have been made to incorporate reinforcement learning into the visual tracking problem. To deal with the problem of erroneous update of target's appearance model during online tracking, Choi et al. [6] use RL to learn a policy that can select the appropriate template. The ADNet of Yun et al. [41] tracks the target by making repetitive actions sampled from the action-decision network that is trained with a combination of supervised learning and reinforcement learning. They also use reinforcement learning to explore the potential of training the agent using weakly labelled data. Zhong et al. [43] then extends ADNet to a hierarchical tracking framework by combining it with the Kernelized Correlation Filter [15] and they also adopt the peak-to-sidelobe ratio (PSR) metric to detect potential tracking failures. The DRL-IS [31] integrates an actor-critic framework into visual tracking to iteratively predict the target's shift. Instead of performing repetitive actions, Chen et al. [5] propose to predict the target's state in a continuous space where only one action is taken for each video frame. They focus on getting good tracking accuracy (exploitation) only but ignore the problem of getting stuck on the non-target objects (exploration).

3 Proposed Method

We firstly outline the correlation filter for visual tracking and then describe how our tracking problem is formulated within the reinforcement learning framework. Next, the training process and the implementation details are given. The standard notations for reinforcement learning [1, 13, 22] are used where possible.

3.1 A Revisit of Correlation Filter Tracking

In correlation-based visual tracking, given an image patch $\mathbf{X}$ that denotes the target to be tracked, our goal is to find the weight matrix $\mathbf{W}$ such that

$$\|\mathbf{W}\mathbf{X} - \mathbf{Y}\| + \lambda\|\mathbf{W}\| \tag{1}$$

is minimised. Here λ is a regularisation coefficient and $\mathbf{Y}$ is the desired regression response when $\mathbf{X}$ is auto-correlated with itself. In the 2D image space, $\mathbf{Y}$ is often modelled as a 2D Gaussian centred at the centroid of $\mathbf{X}$. It is straightforward to verify that the Fourier transform of $\mathbf{W}$ can be computed as follows:

$$\hat{\mathbf{W}} = \frac{\hat{\mathbf{Y}}\hat{\mathbf{X}}}{\hat{\mathbf{X}}^{*}\hat{\mathbf{X}} + \lambda}, \tag{2}$$

where superscript * denotes the complex conjugate, and $\hat{}$ denotes the Fourier transform.

In a new video frame where the tracked target has moved slightly to a different location and its appearance may have changed slightly also, the response map $\mathbf{M}$ of the candidate image patch $\mathbf{Z}$ can be computed via the inverse Fourier transform $\mathcal{F}^{-1}$ as follows:

$$\mathbf{M} = \mathcal{F}^{-1}\left(\hat{\mathbf{W}}\hat{\mathbf{Z}}\right). \tag{3}$$

It can be easily verified that if $\mathbf{Z} = \mathbf{X}$ (i.e., the tracked target is stationary and there is no change in appearance) then $\mathbf{M} = \mathbf{Y}$ and the pixel coordinates giving the maximum response will remain at the same centroid. Otherwise, the pixel location having the maximum correlation response is the translated centroid of the tracked target in the candidate image patch $\mathbf{Z}$. The $\mathbf{Z}$ and $\mathbf{X}$ image patches described above are both high-dimensional feature maps, e.g., computed from a convolutional neural network.

3.2 Maximum Entropy Reinforced Tracker

The aim of a visual tracker is to estimate the target's size and position in each video frame after the ground truth information of the target is given in the first frame. This typical problem of making sequential decisions can be considered as a Markov Decision Process (MDP). The basic components of an MDP include a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state transition function $\mathcal{T}$ and a reward function R. In

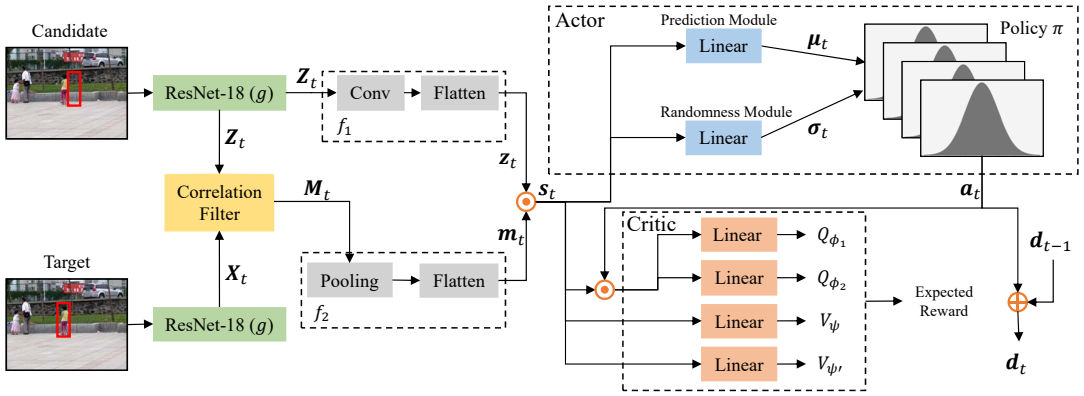


Figure 1: The training pipeline of the proposed method. Both the candidate and the target are passed to ResNet-18 for feature extraction. The correlation filter takes the features as input and produces the response map. The Actor tracks the target using the policy generated by the prediction and the randomness modules. The Critic evaluates the action taken by the Actor and output expected reward for the network parameter update. $\odot$ denotes concatenation and $\oplus$ means element-wise addition.

our MDP formulation, the visual tracker is the agent, which makes sequential decisions (actions) in every video frame to track the target. In frame t, using the known state $\mathbf{s}_t \in \mathcal{S}$, the agent needs to find an optimal action $\mathbf{a}_t \in \mathcal{A}$ which maximises the reward r_t and yields the state $\mathbf{s}_{t+1}$ computed via state transition function $\mathcal{T}$. Our pipeline is end-to-end, taking a video and the target bounding box coordinates in its first frame as input and producing the bounding box coordinates for each video frame as outputs. Being end-to-end, it means that our pipeline is easy to train, optimise, and reason with. The detailed settings of these components are given in the subsections below.

3.2.1 State

To capture both the target's appearance and the response of the interaction with the background, we define the state $\mathbf{s}_t \in \mathcal{S}$ to consist of two parts:

- the processed feature representation $\mathbf{z}_t$ of the candidate;
- the processed response map $\mathbf{m}_t$ calculated by the correlation filter.

We denote the target bounding box in frame t by $\mathbf{d}_t = (x_t, y_t, x'_t, y'_t)$, where (x_t, y_t) and (x'_t, y'_t) represent its top-left and bottom-right corners. Vector $\mathbf{d}_t$ is either the ground truth bounding box given in the first frame ($t = 1$) or the optimal bounding box estimated by our tracker ($t \geqslant 2$). Using the bounding box $\mathbf{d}_{t-1}$ from frame $t-1$ and the image frame $\mathbf{I}_t$, the feature $\mathbf{z}_t$ is computed as follows:

$$\mathbf{z}_t = f_1(\mathbf{Z}_t) = f_1(g(\mathbf{d}_{t-1}, \mathbf{I}_t)), \tag{4}$$

where $\mathbf{Z}_t = g(\mathbf{d}_{t-1}, \mathbf{I}_t)$ is the 2D feature map of the candidate, $f_1(\cdot)$ produces a flattened vector of the input feature map after it has passed through a convolutional layer. As the correlation filter requires sufficient background information around the bounding box of the tracked object, the g function handles the feature map computation after the bounding box cropped from the input frame $\mathbf{I}_t$ has been padded with background pixels.

In the training phase, the candidate feature map $\mathbf{Z}_t$ is compared with the ground truth feature map $\mathbf{X}_t = g(\mathbf{G}_t, \mathbf{I}_t)$, where $\mathbf{G}_t = (\bar{x}_t, \bar{y}_t, \bar{x}'_t, \bar{y}'_t)$ denotes the ground truth bounding box. So the response map $\mathbf{M}_t$ can be computed using $\mathbf{X}_t$ and $\mathbf{Z}_t$ (Eq. (3)). The vector $\mathbf{m}_t$ is produced by downsampling (average pooling) the response map $\mathbf{M}_t$ and then a flattening operation via f_2, i.e.,

$$\mathbf{m}_t = f_2(\mathbf{M}_t). \tag{5}$$

Finally, the state $\mathbf{s}_t$ is defined as

$$\mathbf{s}_t = \text{concat}(\mathbf{z}_t, \mathbf{m}_t). \tag{6}$$

3.2.2 Action

In frame t, the tracking agent samples one action $\mathbf{a}_t$ according to the state $\mathbf{s}_t$ using a suitable tracking policy π, i.e., $\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)$. We define the action as $\mathbf{a}_t = (\Delta x_t, \Delta y_t, \Delta x'_t, \Delta y'_t)$ which describes the translation of the target estimated by the tracking agent. The optimal bounding box $\mathbf{d}_t$ of the target is then calculated as

$$\mathbf{d}_t = \mathbf{d}_{t-1} + \mathbf{a}_t. \tag{7}$$

3.2.3 State Transition

The state transition $\mathcal{T}$ for the tracking agent is more complex than the reinforcement learning framework in game playing as the state and action spaces are both continuous. In a nutshell, once the bounding box $\mathbf{d}_t$ is obtained from $\mathbf{a}_t$ using (7), it can be used for the next frame $\mathbf{I}_{t+1}$ to yield $\mathbf{z}_{t+1}$ and $\mathbf{m}_{t+1}$ using (4)-(5) and so on.

3.2.4 Reward

In reinforcement learning, the reward at time t is a measure of how good an action is. In the visual tracking setting, the reward r_t depends on how good the bounding box $\mathbf{d}_t$ (as a result of taking action $\mathbf{a}_t$) is, i.e., whether it coincides well with the ground truth bounding box. So r_t can be defined as

$$r_t = \begin{cases} \text{IoU}(\mathbf{d}_t, \mathbf{G}_t), & \text{if IoU}(\mathbf{d}_t, \mathbf{G}_t) > \tau \\ -1, & \text{otherwise} \end{cases} \tag{8}$$

where $\text{IoU}(\cdot)$ returns the *intersection-over-union* score of two bounding boxes and τ is a threshold. In (8), a negative reward may result if the IoU value is too small, indicating that the tracker is losing (or has lost) the target.

3.2.5 Training Pipeline

The training pipeline of the proposed method is shown in Figure 1 (see also the pseudocode given in the algorithm). Our tracking agent is trained under the Soft Actor-Critic (SAC) framework proposed by Haarnoja et al. [13]. In this framework, the Actor controls how the

tracking agent behaves based on a policy π while the Critic, consisting of the Q-functions and V-functions, measures how good the action taken is.

The training process here is similar to game playing. In each iteration, a video is randomly selected from the training set. The ground truth bounding box $\mathbf{d}_1$ is given in the first video frame $\mathbf{I}_1$ to initialise the visual tracker and the tracking process starts from frame 2. To be consistent with the terminology used in visual tracking, we denote the initial state in our method as $\mathbf{s}_2$ rather than $\mathbf{s}_0$ which is commonly used as the initial state in reinforcement learning.

Given $\mathbf{d}_1$ and $\mathbf{I}_1$, the ground truth feature map $\mathbf{X}_1$ is obtained from $g(\mathbf{G}_1, \mathbf{I}_1)$. The correlation filter is then initialised using (2) which sets up the variable $\hat{\mathbf{W}}$. From the video frame $t \geqslant 2$, using the input frame $\mathbf{I}_t$, vectors $\mathbf{z}_t$ and $\mathbf{m}_t$ are computed using (4) and (5); state vector $\mathbf{s}_t$ is then obtained from (6). The policy π is modelled as a 4-dimensional Gaussian distribution with mean $\boldsymbol{\mu}_t$ and covariance matrix diag($\boldsymbol{\sigma}_t$), where $\boldsymbol{\mu}_t \in \mathbb{R}^4$ is produced by the *prediction module* and $\boldsymbol{\sigma}_t \in \mathbb{R}^4$ is produced by the *randomness module*. Both modules are modelled as linear layers in the pipeline (Figure 1). In the training phase, the ground truth feature maps $\mathbf{X}_t = g(\mathbf{G}_t, \mathbf{I}_t), \forall t \geqslant 2$, are used to guide the tracking process which in turn helps to optimise the policy π.

The tracking agent keeps interacting with the environment until the tracking process terminates because the end of the video is reached or when the target is lost. We use a binary variable p to denote whether the tracking process is still active. When $p = 0$, it means that the tracking process is still on-going; when $p = 1$, it means that the tracking process has stopped, equivalent to "game over" in playing a game. We define $b_t = (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, p_{t+1})$ as the training sample for video frame t. However, these training samples are not independently and identically distributed because they are sequentially generated. They are therefore not suitable to use for training neural networks in the proposed method. To deal with this problem, we adopt the same "replay buffer" trick used by Lillicrap et al. [22] to store the training samples for all t. So the whole tracking process can be explicitly represented by a replay buffer $\mathcal{B} = (b_2, \cdots, b_{T-1})$ where T is the number of frames in the current video or before tracking terminates, whichever is smaller. After all the training samples are collected in $\mathcal{B}$, our algorithm then samples uniformly from $\mathcal{B}$ to train the neural networks.

3.2.6 Network Parameter Update

For the actor model, the goal is to learn the optimal policy π that maximises the expected reward (tracking accuracy) and the entropy at the same time. Different from the traditional reinforcement learning, the optimal policy π^* in our maximum entropy RL framework is defined as

$$\pi^* = \arg\max_{\pi} \mathrm{E}\left[\sum_{t=0}^{T} \gamma^t \left(R\left(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}\right) + \alpha H\left(\pi\left(\cdot|\mathbf{s}_t\right)\right)\right)\right], \quad (9)$$

where $\mathrm{E}[\cdot]$ denotes the expectation, $0 < \gamma < 1$ is the discounted factor and $H(\cdot)$ denotes the entropy. In order to learn this optimal policy π^*, we follow the setting in SAC [13] where our algorithm concurrently learns a policy π, two Q-functions Q_{ϕ_1} and Q_{ϕ_2}, and a value function V_ψ, where ϕ_1, ϕ_2 and ψ are the corresponding parameters in these functions. In this setting, rather than the Actor learns to evaluate the expected reward, a Critic module is used to learn the complex latent information (captured in Q_{ϕ_1}, Q_{ϕ_2}, and V_ψ) that leads to optimal long-term reward.

To improve the stability of the training process, we adopt the trick of *goal value function*[1] that is commonly used in reinforcement learning [22, 13, 25], i.e., a separate goal value function $V_{\psi'}$ with fixed parameter copied over from the value function V_ψ is used and is updated during training.

Algorithm Training Pipeline

Input: Training videos with ground truth bounding boxes of the target; N_{epoch}: number of training epochs (default value 5000); N_{iter}: number of updating iterations (default value 1000)
Output: Trained network parameters for optimal policy
Initialise Q-function parameters ϕ_1, ϕ_2, V-function parameter ψ and goal value function parameter ψ'.
for $0 \leqslant i \leqslant N_{\text{epoch}}$ **do**
  $\mathcal{B} \leftarrow \varnothing$
  Randomly choose one video from the training set
  Initialise the correlation filter using $\mathbf{I}_1$ and $\mathbf{d}_1$.
  Compute initial state $\mathbf{s}_2$ using $\mathbf{I}_2$ and $\mathbf{d}_1$
  for $2 \leqslant t \leqslant T-1$ **do**
    Get action $\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)$
    Execute $\mathbf{a}_t$ to get the new bounding box $\mathbf{d}_t$
    Compute next state $\mathbf{s}_{t+1}$ using (4), (5), (6).
    Compute reward r_t using (8).
    if the target is lost or the video is finished **then**
      $p_{t+1} \leftarrow 1$
    end if
    Store $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, p_{t+1})$ in the set $\mathcal{B}$
    if $p_{t+1} = 1$ **then**
      for $0 \leqslant j \leqslant N_{\text{iter}}$ **do**
        Randomly choose N_b samples from $\mathcal{B}$
        Update ϕ_i in $\mathcal{L}_{Q_i}$ using (10)
        Update ψ in $\mathcal{L}_V$ using (11)
        Update parameters in $\mathcal{L}_\pi$ using (12)
        Update ψ' with (13)
      end for
      break
    end if
  end for
end for

For each step in the update stage, N_b training samples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}, p_{t+1})$ are randomly drawn from $\mathcal{B}$. The updating of Q-functions and V-function can be achieved by minimising the losses $\mathcal{L}_{Q_i}$ and $\mathcal{L}_V$ [13] defined below:

$$\mathcal{L}_{Q_i} = \frac{1}{N_b} \sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{B}} \left(Q_{\phi_i}(\mathbf{s}_t, \mathbf{a}_t) - \kappa_t^q\right)^2 \quad (10)$$

$$\mathcal{L}_V = \frac{1}{N_b} \sum_{\mathbf{s}_t \in \mathcal{B}} \left(V_\psi(\mathbf{s}_t) - \kappa_t^v\right)^2, \quad (11)$$

where $\kappa_t^q = r_t + \gamma(1 - p_{t+1}) V_{\psi'}(\mathbf{s}_{t+1})$ and $\kappa_t^v = \min_{i=1,2} Q_{\phi_i}(\mathbf{s}_t, \tilde{\mathbf{a}}_t) - \alpha \log \pi(\tilde{\mathbf{a}}_t|\mathbf{s}_t)$.

Since our aim is to maximise the overall reward, the policy is updated using gradient descent to optimise (see [13]):

$$\mathcal{L}_\pi = \frac{1}{N_b} \sum_{\mathbf{s}_t \in B} \left(Q_{\phi_1}(\mathbf{s}_t, \tilde{\mathbf{a}}_t) - \alpha \log \pi(\tilde{\mathbf{a}}_t|\mathbf{s}_t)\right). \quad (12)$$

Finally, the parameter of the goal value function ψ' is updated with incremental learning:

$$\psi' \leftarrow \rho\psi' + (1 - \rho)\psi, \quad (13)$$

[1] We use the phrase *goal value function* instead of *target value function* to avoid confusion with the target being tracked.

Parameter	θ	λ	τ	γ	α	ρ
Value	0.01	0.01	0.5	0.99	0.2	0.995

Table 1: Parameter values used in the proposed method.

where $\rho \in (0, 1)$ is the update rate.

3.2.7 *Online Tracking*

This is the testing phase of the algorithm. After the Actor which holds the policy π has been trained, online tracking can be carried out. The online tracking steps are similar to those described under the *Training Pipeline* subsection except for the following:

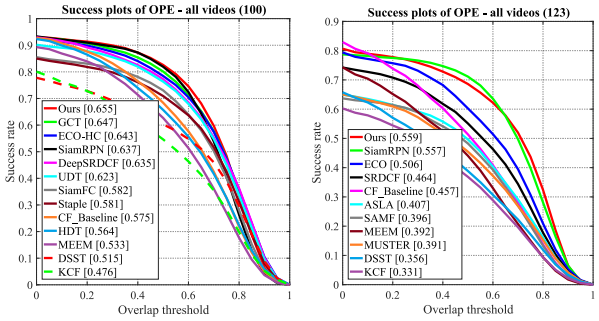
- The reward r_t does not need to be computed. This means that the Critic is not required in online tracking (its role of helping the Actor to generate optimal policy by optimising the expected reward has been fulfilled).
- Ground truth bounding boxes are no longer available, except for the first video frame. So, in Figure 1, $\mathbf{X}_1 = g(\mathbf{G}_1, \mathbf{I}_1)$ as before, but $\mathbf{X}_t, \forall t \geqslant 2$, is the target model obtained from the incremental learning in (14) below.
- The feature map $\mathbf{X}_t$, for each $t \geqslant 2$, that represents the target model is updated with incremental learning via
$$\mathbf{X}_t = (1 - \theta)\mathbf{X}_{t-1} + \theta\mathbf{X}'_t, \tag{14}$$
where $0 < \theta < 1$ denotes the learning rate and $\mathbf{X}'_t = g(\mathbf{d}_t, \mathbf{I}_t)$ is the optimal feature map obtained from the computed $\mathbf{d}_t$. Variable $\hat{\mathbf{W}}$ is computed accordingly for $\mathbf{X}_t$ using (2) for each video frame.

Lastly, the optimal bounding box $\mathbf{d}_t$ output by our method is compared against the ground truth for each frame t.

3.2.8 *Implementation Details*

The values of some parameters in our method are shown in Table 1. We adopt a ResNet-18 [14] as the feature extractor g in (4). So the dimensions of both $\mathbf{Z}_t$ and $\mathbf{X}_t$ are $18 \times 18 \times 256$. The response map $\mathbf{M}_t$ is resized to 288×288. The candidate pre-process function f_1 in (4) includes a 1×1 convolutional layer of 256 input channels and 1 output channel followed by a "flatten" operation. The response map pre-process function f_2 in (5) includes an average pooling layer with a filter size of 4 and a "flatten" operation. So the dimensions of $\mathbf{z}_t$, $\mathbf{m}_t$ and $\mathbf{s}_t$ are 324, 5184, and $324+5184 = 5508$ respectively. The prediction module takes $\mathbf{z}_t$ as input and outputs $\boldsymbol{\mu}_t$, while the input and output of the randomness module are $\mathbf{m}_t$ and $\boldsymbol{\sigma}_t$. The linear layers of these two modules output 4 neurons and have 324 and 5184 input neurons respectively. The Q_{ϕ_1}, Q_{ϕ_2}, V_ψ, and $V_{\psi'}$ functions are all modelled as single linear layers having 5512, 5512, 5508, and 5508 input neurons respectively and 1 output neuron. 5000 epochs (line 4 of pseudocode) for network training and 1000 iterations (line 19 of pseudocode) for updating the value function were found to be sufficient based on our experiments. The parameter values used in Table 1 were obtained from our hyperparameter search process.

Our method is implemented with PyTorch. In the online tracking phase, our method is able to achieve real-time performance, processing 50 frames per second on a desktop with Intel i7 CPU and TITAN Xp GPU.



(a) OTB100 (b) UAV123

Figure 2: Success plots on OTB100 and UAV123 for comparison with state-of-the-art methods. Larger area under curve (AUC) values (number inside square brackets) mean better algorithms.

4 Experiments

4.1 Datasets

We evaluate the proposed method on 5 datasets: OTB100 [40], UAV123 [27], LaSOT [10], VOT2016 [17] and VOT2018 [18], which includes 100, 123, 280, 60 and 60 videos respectively. These benchmarks have been widely used by the visual tracking community. Almost all the challenging factors in visual tracking such as illumination change, occlusion and deformation can be found in the videos from these datasets.

For the OTB100, UAV123 and LaSOT datasets, the One Pass Evaluation (OPE) is used. In OPE, the tracker is initialised at the beginning of the video and tracking is performed till the end of the video. This is different from the "re-start" setting in the VOT datasets where the evaluation toolkit automatically resets the tracker if a tracking failure is detected (when the IoU score [39] between the ground truth and the tracking result is zero).

4.2 Training Data

LaSOT has 1400 videos labelled with 14 different attributes: Illumination Variation (IV), Partial Occlusion (PO), Deformation (DEF), Motion Blur (MB), Camera Motion (CM), Rotation (RO), Background Clutter (BC), Viewpoint Change (VC), Scale Variation (SV), Full Occlusion (FO), Fast Motion (FM), Out-of-View (OV), Low Resolution (LR), and Aspect Ratio Change (ARC). It is split into a training set (having 1120 videos) and a test set (280 videos). Our model is trained using the training set and the reported results of our method on LaSOT are on the test set. The same trained model is also used to test on other datasets reported in this paper.

4.3 Evaluation Metrics

For the OTB100, UAV123 and LaSOT (test set) datasets, we use the success plot to evaluate our tracking algorithm. The success plot measures the proportion of tracking successes against the IoU threshold τ (see (8)). So when τ is small, most bounding boxes are deemed to be good and the success rate is high. The area under curve (AUC) is commonly used for comparing the performance of different tracking algorithms. For the VOT datasets [17, 18], we adopt the metrics *accuracy*, *robustness* and *expected average overlap* (EAO) described in their publications as the evaluation metrics.

	CFNet	CSRDCF	ECO	fDSST	PTAV	STRCF	MDNet	VITAL	TRACA	CF_Baseline	Ours
All	0.2746	0.2444	0.3241	0.2032	0.2503	0.3084	0.3970	0.3898	0.2575	0.3326	**0.4106**
IV	0.3180	0.2807	0.3733	0.2479	0.2660	0.3342	0.4074	0.4034	0.2936	0.2992	**0.4441**
PO	0.2464	0.2112	0.2900	0.1746	0.2171	0.2732	0.3699	0.3612	0.2398	0.3117	**0.3932**
DEF	0.2714	0.2349	0.2789	0.1621	0.2081	0.2738	0.3910	0.3841	0.2297	0.3137	**0.4097**
MB	0.2375	0.2261	0.3052	0.1926	0.2441	0.3238	0.3757	0.3631	0.2450	0.3137	**0.3971**
CM	0.2886	0.2506	0.3576	0.2024	0.2655	0.3436	0.4157	0.3974	0.2917	0.3303	**0.4427**
RO	0.2468	0.2213	0.2847	0.1718	0.2186	0.2647	0.3787	0.3706	0.2215	0.3258	**0.3978**
BC	0.2714	0.2329	0.3188	0.2216	0.2512	0.3167	**0.3739**	0.3646	0.2777	0.3070	0.3605
VC	0.2476	0.2376	0.3166	0.1364	0.2030	0.2832	0.3579	0.3386	0.2076	0.2752	**0.3656**
SV	0.2672	0.2394	0.3178	0.1951	0.2414	0.3028	0.3924	0.3850	0.2517	0.3249	**0.4056**
FO	0.1989	0.1666	0.2543	0.1423	0.1792	0.2345	0.3046	0.3010	0.2014	0.2947	**0.3367**
FM	0.1558	0.1398	0.2334	0.1176	0.1681	0.1977	0.2599	0.2465	0.1543	0.2768	**0.3326**
OV	0.1827	0.1754	0.2389	0.1373	0.1757	0.2315	0.3297	0.3038	0.1726	0.2844	**0.3513**
LR	0.1949	0.1772	0.2674	0.1470	0.1979	0.2450	0.3170	0.3086	0.1840	0.2910	**0.3270**
ARC	0.2430	0.2152	0.2884	0.1675	0.2119	0.2688	0.3664	0.3581	0.2270	0.3156	**0.3961**

Table 2: Experimental results on the LaSOT dataset. Results of other trackers are extracted from the authors' papers.

	ADNet	RLS-CFV	ACT	DRL-IS	RDT	Ours
Implementation	MATLAB	Python	Python	Python	Python	Python
CPU/GPU	Core i7 / TITAN X	– / GTX 1060	– / TITAN	GTX 1080 Ti	Core i7 / TITAN X	Core i7 / TITAN Xp
Frame rate	2.9	6.3	30	22	43	**50**
AUC (OTB100)	0.646	0.651	0.643	0.593	0.603	**0.655**
Year	2017	2018	2018	2018	2018	–

Table 3: Comparison with visual trackers based on reinforcement learning on the OTB100 dataset. The information for these trackers is extracted from the authors' publications.

α	0	0.1	0.2	0.3	0.5	0.7	0.9
AUC	0.610	0.620	0.655	0.645	0.637	0.623	0.619

Table 4: Evaluation of the maximum entropy on OTB100 dataset.

4.4 Ablation Study

4.4.1 Compared with Correlation Filter Baseline

As our tracker uses the output from the correlation filter to estimate the target's translation, we consider the correlation filter tracking as our baseline. That is, the translation of the target is obtained from the response map $\mathbf{M}$ in (3) directly without involving the Actor-Critic part. We evaluate both our method and the baseline on the OTB100, UAV123 and LaSOT (test set) datasets. As shown in Figure 2(a), without the tracking agent trained using reinforcement learning, the CF_Baseline method only achieves 57.5% in AUC while the proposed method gains 8% improvement and achieves 65.5%. The proposed method also outperforms CF_Baseline by 10.2% in AUC on the UAV123 dataset (see Figure 2(b)). A gain of 7.8% in overall AUC value compared to the baseline is also achieved by the proposed method on LaSOT shown in Table 2.

4.4.2 Evaluation of Maximum Entropy

The parameter α in (9) indicates how much influences the entropy term has on our algorithm. To demonstrate the effectiveness of "maximum entropy", the proposed method is tested against different α values on the OTB 100 dataset. As shown in Table 4, when the value of α increases, the AUC value first increases as well and then drops accordingly. While the best performance (AUC value) is achieved when $\alpha = 0.2$, the baseline of reinforcement learning tracking without maximum entropy is when $\alpha = 0$, which gives an AUC value of 61.0%. This demonstrates that the proposed maximum entropy reinforced tracker has improved a lot compared to the RL-based tracker baseline.

4.5 Comparison with Visual Trackers Based on RL

Using the OTB100 dataset, we compare our proposed method with other reinforcement learning based visual trackers including ADNet [41], RLS-CFV [43], ACT [5], DRL-IS [31] and RDT [6]. To the best of our knowledge, our method is the first that combines visual tracking with maximum entropy reinforcement learning. Table 3 shows the AUC values and processing speeds of these six methods. Our method achieves the highest AUC value (65.5%), leading the runner-up RLS-CFV by a small margin. The frame rates shown in the table serve as a guideline only as the methods were not implemented on the same hardware. However, it shows that with a mid-range GPU, our method can already achieve a frame rate of 50 (for pre-recorded videos).

4.6 Comparison with state-of-the-art methods

We compare our tracker with state-of-the-art methods on five challenging tracking benchmarks. All the results of other trackers are extracted from either the authors' papers or from the public benchmarks. Depending on the availability of the tracking results, different methods are compared for different datasets.

4.6.1 OTB100

The videos in OTB100 are labelled with 11 attributes. Figure 2(a) shows the results of GCT [12], ECO-HC [8], SiamRPN [19], DeepSRDCF [7], UDT [37], SiamFC [3], Staple [2], HDT [29], DSST [9], KCF [15], and our tracker on the OTB100 dataset. Our tracker achieves the highest AUC value of 65.5% and gives an improvement of 0.8% compared to the second place GCT.

4.6.2 UAV123

Besides some classic trackers such as KCF [15], DSST [9], and ECO [8], the recent state-of-the-art method SiamRPN [19] is also added to the comparison. Figure 2(b) shows that our proposed

Datasets	Trackers	EAO ↑	Robustness ↓	Accuracy ↑
VOT2016	EBT (2016)	0.291	0.252	0.465
	Staple (2016)	0.295	0.378	0.544
	TCNN (2016)	0.325	0.268	0.554
	DNT (2017)	0.278	0.329	0.515
	SA-Siam (2018)	0.291	-	0.540
	SiamRPN (2018)	0.344	-	0.560
	TADT (2019)	0.299	-	0.550
	Ours	**0.361**	**0.242**	**0.623**
VOT2018	ECO (2017)	0.280	0.276	0.484
	DCFNet (2017)	0.182	0.543	0.470
	SiamFC (2016)	0.188	0.585	0.503
	SRCF (2018)	0.310	0.290	0.520
	CFTR (2018)	0.300	**0.258**	0.505
	SiamVGG (2018)	0.286	0.318	0.531
	CSRDCF (2017)	0.263	0.318	0.466
	SiamRPN (2018)	**0.383**	0.276	0.586
	Ours	0.322	0.295	**0.589**

Table 5: Results on VOT2016 and VOT2018. The up arrow ↑ means the higher the better and vice versa.

method achieves an AUC score of 55.9%, which is slightly higher than the second-place SiamRPN, and outperforms ECO (50.6%) by a large margin.

4.6.3 *LaSOT*

Table 2 shows the results of CFNet [36], CSRDCF [23], ECO [8], DSST [9], PTAV [11], MDNet [28], VITAL [33], and the proposed method. Our method achieves the best performances on all the categories except for *Rotation* (RO). Our method obtains an overall 41.06% AUC value, which is 1.36% higher than the second place MDNet. For the *Rotation* category, our AUC value is 36.05%, which is 1.34% lower than the first place MDNet. This slightly poorer performance of our method is because the features extracted from the convolutional neural network are not rotation-invariant. To generate augmented data where the target is rotated for training might help alleviate the effect of rotation.

4.6.4 *VOT2016*

Following the evaluation protocol of VOT2016, We test the proposed method on all the videos. The comparison results are given in Table 5. Our method achieves the best performance on EAO, *accuracy* and *robustness*. Compared with the latest method TADT [20], our method gains a 0.062 improvement of EAO. Our method also outperforms the current state-of-the-art method SiamRPN with a gain of 0.017 of EAO.

4.6.5 *VOT2018*

The VOT2018 dataset contains 60 videos and has more accurate ground truth bounding box compared to VOT2016. The evaluation criterion in VOT2018 is the same as that in VOT2016, i.e., the tracker is reset if the target is lost. In Table 5, we compare our method with both correlation filter tracers and siamese trackers including ECO [8] and SiamRPN [19]. Our method achieves the best performance in terms of EAO (0.322) and *accuracy* (0.589), while maintaining a competitive *robustness* score (0.295).

4.7 Qualitative Evaluation

The visual comparison between the proposed method and some representative trackers including ECO-HC, DeepSRDCF, SiamRPN and

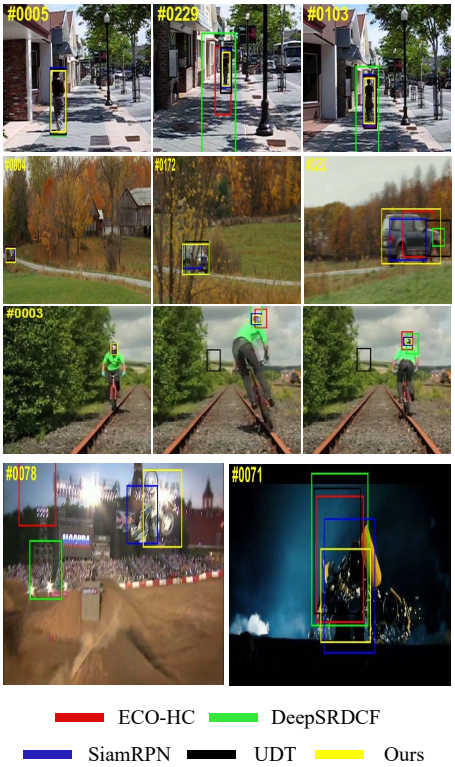


Figure 3: Screenshots from 3 challenging videos (top 3 rows) of OTB100: *Human9*, *CarScale*, and *Bike1*. Two failure cases are from videos *MotorRolling* and *Trans* (bottom row) of OTB100.

UDT on three videos taken from the OTB100 dataset is shown at the first three rows of Figure 3. For videos *Human9* and *carScale*, the targets have very large scale variations. Our tracker can handle this problem well and always gives precise bounding boxes. For the video *Bike1*, the target (the biker's head) moves very fast while the biker jumps from left to right. Our tracker is still able to follow the target while other trackers either lose the target or give much larger bounding box than the target's real size.

4.8 Failure Case Analysis

Two failure cases are shown at the bottom row of Figure 3. In the video *MotorRolling*, the motor rider rotates 360 degrees. As mentioned above, the features extracted from CNNs is not rotation-invariant. The screenshot shows that all the trackers fail to track the target in this video as they all use CNNs for feature extraction. In the video *Trans*, the target changes from a skinny and tall robot into a vehicle on the ground. This kind of fast deformation is very difficult for visual trackers to follow so the tracking failure is not unexpected.

5 Conclusion

In this paper, we have made the first attempt to combine visual tracking and maximum entropy reinforcement learning, to help our visual tracker avoid getting stuck on the wrong targets while maintaining high tracking accuracy. Our method has been evaluated on several popular benchmarks and compared with state-of-the-art methods to demonstrate its tracking accuracy and real-time processing speed. Our future work will focus on how to deal with rotation as well as fast deformation of targets.

REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath, 'Deep Reinforcement Learning: A Brief Survey', *IEEE Signal Processing Magazine*, **34**(6), 26–38, (2017).

[2] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr, 'Staple: Complementary Learners for Real-Time Tracking', in *CVPR*, pp. 1401–1409, (2016).

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr, 'Fully-Convolutional Siamese Networks for Object Tracking', in *ECCV*, pp. 850–865. Springer, (2016).

[4] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, 'Visual Object Tracking using Adaptive Correlation Filters', in *CVPR*, pp. 2544–2550, (June 2010).

[5] Boyu Chen, Dong Wang, Peixia Li, Shuang Wang, and Huchuan Lu, 'Real-time 'Actor-Critic' Tracking', in *ECCV*, pp. 318–334, (2018).

[6] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee, 'Real-time Visual Tracking by Deep Reinforced Decision Making', *Computer Vision and Image Understanding*, **171**, 10–19, (2018).

[7] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, 'Learning Spatially Regularized Correlation Filters for Visual Tracking', in *ICCV*, pp. 4310–4318, (Dec 2015).

[8] Martin Danelljan, Goutam Bhat, F Shahbaz Khan, and Michael Felsberg, 'ECO: Efficient Convolution Operators for Tracking', in *CVPR*, pp. 21–26, (2017).

[9] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg, 'Accurate Scale Estimation for Robust Visual Tracking', in *BMVC*, (2014).

[10] Heng Fan, Liting Lin, Fan Yang, Peng Chu, Ge Deng, Sijia Yu, Hexin Bai, Yong Xu, Chunyuan Liao, and Haibin Ling, 'LaSOT: A High-Quality Benchmark for Large-Scale Single Object Tracking', in *CVPR*, pp. 5374–5383, (2019).

[11] Heng Fan and Haibin Ling, 'Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking', *arXiv:1708.00153*, (2017).

[12] Junyu Gao, Tianzhu Zhang, and Changsheng Xu, 'Graph Convolutional Tracking', in *CVPR*, pp. 4649–4659, (2019).

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor', *arXiv:1801.01290*, (2018).

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep Residual Learning for Image Recognition', in *CVPR*, pp. 770–778, (2016).

[15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, 'High-Speed Tracking with Kernelized Correlation Filters', *IEEE TPAMI*, **37**(3), 583–596, (March 2015).

[16] Jianglei Huang and Wengang Zhou, 'Re2EMA: Regularized and Reinitialized Exponential Moving Average for Target Model Update in Object Tracking'. AAAI, (2019).

[17] Matej Kristan, Aleš Leonardis, Jiři Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin, and et al, *The Visual Object Tracking VOT2016 Challenge Results*, 777–823, Cham, 2016.

[18] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Cehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al., 'The Sixth Visual Object Tracking VOT2018 Challenge Results', in *ECCV*, (2018).

[19] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu, 'High Performance Visual Tracking with Siamese Region Proposal Network', in *CVPR*, pp. 8971–8980, (2018).

[20] Xin Li, Chao Ma, Baoyuan Wu, Zhenyu He, and Ming-Hsuan Yang, 'Target-Aware Deep Tracking', in *CVPR*, pp. 1369–1378, (2019).

[21] Yang Li, Jianke Zhu, Steven CH Hoi, Wenjie Song, Zhefeng Wang, and Hantang Liu, 'Robust Estimation of Similarity Transformation for Visual Object Tracking', in *AAAI*, volume 33, pp. 8666–8673, (2019).

[22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, 'Continuous Control with Deep Reinforcement Learning', *arXiv preprint arXiv:1509.02971*, (2015).

[23] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan, 'Discriminative Correlation Filter with Channel and Spatial Reliability', in *CVPR*, pp. 6309–6318, (2017).

[24] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu, 'Reinforcement Learning for Visual Object Detection', in *CVPR*, pp. 2894–2902, (2016).

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, 'Playing Atari with Deep Reinforcement Learning', *arXiv preprint arXiv:1312.5602*, (2013).

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., 'Human-Level Control through Deep Reinforcement Learning', *Nature*, **518**(7540), 529, (2015).

[27] Matthias Mueller, Neil Smith, and Bernard Ghanem, 'A Benchmark and Simulator for UAV Tracking', in *ECCV*, pp. 445–461. Springer, (2016).

[28] Hyeonseob Nam and Bohyung Han, 'Learning Multi-Domain Convolutional Neural Networks for Visual Tracking', in *CVPR*, pp. 4293–4302, (2016).

[29] Yuankai Qi, Shengping Zhang, Lei Qin, Hongxun Yao, Qingming Huang, Jongwoo Lim, and Ming-Hsuan Yang, 'Hedged Deep Tracking', in *CVPR*, pp. 4303–4311, (2016).

[30] Yuankai Qi, Shengping Zhang, Weigang Zhang, Li Su, Qingming Huang, and Ming-Hsuan Yang, 'Learning Attribute-Specific Representations for Visual Tracking', in *AAAI*, volume 33, pp. 8835–8842, (2019).

[31] Liangliang Ren, Xin Yuan, Jiwen Lu, Ming Yang, and Jie Zhou, 'Deep Reinforcement Learning with Iterative Shift for Visual Tracking', in *ECCV*, pp. 684–700, (2018).

[32] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller, 'Deterministic Policy Gradient Algorithms', in *ICML*, (2014).

[33] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang, 'VITAL: Visual Tracking via Adversarial Learning', in *CVPR*, pp. 8990–8999, (2018).

[34] Richard S Sutton, Andrew G Barto, et al., *Introduction to Reinforcement Learning*, volume 2, MIT press Cambridge, 1998.

[35] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou, 'Deep Progressive Reinforcement Learning for Skeleton-Based Action Recognition', in *CVPR*, pp. 5323–5332, (2018).

[36] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, 'End-to-End Representation Learning for Correlation Filter Based Tracking', in *CVPR*, pp. 5000–5008, (July 2017).

[37] Ning Wang, Yibing Song, Chao Ma, Wengang Zhou, Wei Liu, and Houqiang Li, 'Unsupervised Deep Tracking', in *CVPR*, pp. 1308–1317, (2019).

[38] Qiang Wang, Jin Gao, Junliang Xing, Mengdan Zhang, and Weiming Hu, 'DCFNet: Discriminant Correlation Filters Network for Visual Tracking', *arXiv:1704.04057*, (2017).

[39] Y. Wu, J. Lim, and M. H. Yang, 'Online Object Tracking: A Benchmark', in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411–2418, (June 2013).

[40] Y. Wu, J. Lim, and M. H. Yang, 'Object Tracking Benchmark', *IEEE TPAMI*, **37**(9), 1834–1848, (Sept 2015).

[41] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi, 'Action-decision Networks for Visual Tracking with Deep Reinforcement Learning', in *CVPR*, pp. 2711–2720, (2017).

[42] Jianfu Zhang, Naiyan Wang, and Liqing Zhang, 'Multi-Shot Pedestrian Re-Identification via Sequential Decision Making', in *CVPR*, pp. 6781–6789, (2018).

[43] Bineng Zhong, Bing Bai, Jun Li, Yulun Zhang, and Yun Fu, 'Hierarchical Tracking by Reinforcement Learning-based Searching and Coarse-to-fine Verifying', *IEEE Transactions on Image Processing*, **28**(5), 2331–2341, (2018).