Visual Tracking by means of Deep Reinforcement Learning and an Expert Demonstrator

Matteo Dunnhofer, Niki Martinel, Gian Luca Foresti and Christian Micheloni
Department of Mathematics, Computer Science and Physics, University of Udine, Italy
dunnhofer.matteo@spes.uniud.it, {niki.martinel, gianluca.foresti, christian.micheloni}@uniud.it

Abstract

In the last decade many different algorithms have been proposed to track a generic object in videos. Their execution on recent large-scale video datasets can produce a great amount of various tracking behaviours. New trends in Reinforcement Learning showed that demonstrations of an expert agent can be efficiently used to speed-up the process of policy learning. Taking inspiration from such works and from the recent applications of Reinforcement Learning to visual tracking, we propose two novel trackers, A3CT, which exploits demonstrations of a state-of-the-art tracker to learn an effective tracking policy, and A3CTD, that takes advantage of the same expert tracker to correct its behaviour during tracking. Through an extensive experimental validation on the GOT-10k, OTB-100, LaSOT, UAV123 and VOT benchmarks, we show that the proposed trackers achieve state-of-the-art performance while running in real-time.

1. Introduction

Visual object tracking is one of the most challenging problems in Computer Vision. In its simplest form, it consists in the persistent recognition and localization –by means of bounding boxes– of a target object in consecutive video frames. Even though many efforts have been recently made (*e.g.* [43, 50, 26]), the process of automatically following a generic object in a video comes with several different challenges including occlusions, light changes, fast motion, and motion blur. In addition, many practical applications of visual tracking, such as video surveillance, behavior understanding, autonomous driving and robotics, require accurate predictions with real-time constraints.

In many current methodologies (*e.g.* [37, 13, 10, 3]), Convolutional Neural Networks (CNNs) [29] pre-trained for image classification showed to be effective for visual tracking. Due to their discriminative power, CNN generated feature representations are widely used to search the target in the consecutive frames. This kind of information is widely used in classification or tracking-by-detection methods (*e.g.* [37, 44]). The most significant drawback of these methods is that they require computational demanding procedures to search for candidate targets in new frames. Furthermore, even strong CNN models may not be able to capture all possible variations of targets and need to be updated online during tracking. In these scenarios, the tracker shall understand the quality of its tracking process and the target's motion status, in order to take decisions to efficiently update its model. Solutions that implement such a mechanism achieve excellent results (*e.g.* [37, 5, 52, 39]), but their processing speed is often far from being real-time. Moreover, the problem of taking decisions online requires algorithms capable of deciding intelligently at the right moments.

To address these issues, tracking methodologies based on Reinforcement Learning (RL) have been recently proposed [4, 52, 19, 3, 39]. The idea behind such works is to treat aspects like target searching procedures or tracking status evaluation as sequential decision-making problems. In these settings, an artificial agent is trained to take optimal sequential decisions to solve a tracking related task which, ultimately, leads to the development of a strategy to track the target object. These solutions maintain competitive performance with state-of-the-art methods, however they implement complex and demanding online update procedures that slow tracking. In addition, these methods are usually not end-to-end and require at least two training stages, one initial supervised learning (SL) stage and a following RL fine-tuning.

We argue that better speed performance can be obtained and that SL can be incorporated into an RL framework to make the training end-to-end. We claim that tracking demonstrations of an expert tracker can be used to guide RL tracking agents. Furthermore, we propose to simplify the online update strategy by taking advantage of the expert during tracking. We will demonstrate that RL functions needed for training can be also used during tracking to exploit the performance of the expert tracker and to consequentially improve the tracking accuracy.

In particular, in this paper we introduce the following contributions:

1. a real-time CNN-based tracker named A3CT which is trained via an end-to-end RL method that takes advantage of the demonstrations of a state-of-the-art tracker;
2. a real-time CNN-based tracker named A3CTD which uses the RL functions learned during training to improve performance by exploiting the expert during the tracking phase.

The proposed trackers are built on a deep regression network for tracking [13, 10] and are trained inside an on-policy Asynchronous Actor-Critic framework [32] that incorporates SL and expert demonstrations. A state-of-the-art tracking algorithm [2] is run on a large-scale tracking dataset [18] to obtain the demonstrations. Experiments will show that the proposed A3CT and A3CTD trackers perform comparably with state-of-the-art methods on the GOT-10k test set [18], LaSOT [9], UAV123 [35], OTB-100 [50] and VOT benchmarks [24, 26], while achieving a processing speed of 90 FPS and 50 FPS respectively.

2. Related work

2.1. Deep RL

RL concerns methodologies to train artificial agents to solve interactive decision-making problems [45]. Recent trends in this field (*e.g*. [33, 34, 41, 42]) showed the successful combination of Deep Neural Networks (DNNs) and RL algorithms (so-called Deep RL) in the representation of models such as the value or policy functions. Among the existing approaches, off-policy strategies aim to learn the state or the state-action value functions, that give estimations about the expected future reward of states and actions [48, 33, 34]. The policy is then extracted by choosing greedily the actions that yield the highest function values. On the other hand, on-policy algorithms directly learn the policy by optimizing the DNN with respect to the expected future reward [49]. There exist then hybrid approaches, known as Actor-Critic [22], that maintain and optimize the model representations of both the policy and state value (or state-action value) functions.

All these methods however suffer of slow convergence, especially in cases where continuous or high-dimensional action spaces are considered. Recent solutions (*e.g*. [47, 36, 40, 20, 15]) propose to use expert demonstrations to help and guide the learning process.

2.2. Visual Tracking

Visual Tracking has received increasing interest thanks to the introduction of new benchmarks [50, 9, 35, 18] and challenges [28, 23, 24, 25, 26].

Thanks to their superior representation power, in recent years various approaches based on CNNs appeared [13, 10, 37, 5, 2, 30, 53]. Held *et al*. [13] and Gordon *et al*. [10] showed how deep regression CNNs could capture the target's motion. However, these methods are trained using SL which optimizes parameters for just local predictions. In contrast, we propose a RL-based training which optimizes the DNN's weights for the maximization of performance in future predictions. Nam *et al*. [37] proposed an online tracking-by-detection approach by using a pre-trained CNN for image classification. Similarly, Danelljan *et al*. [7, 5] proposed a discriminative correlation filter approach by integrating multi-resolution CNN features. These solutions obtained outstanding results w.r.t. the previous methodologies, however they are very computationally expensive and can run at just 1 and 6 FPS respectively. Currently, the approach based on the Siamese framework is getting significant attention for their well-balanced tracking accuracy and efficiency [2, 11, 31, 30, 54, 53]. These trackers formulate the visual tracking as a cross-correlation problem and are leveraging effectively from end-to-end learning of DNNs. However their performance is susceptible to visual distractors due to the non-incorporation of temporal information or online fine-tuning. Conversely to this, our tracker present the use of an LSTM [16] to model the temporal relation of target's appearance between frames.

2.3. Deep RL for Visual Tracking

Very recently, Deep RL has started to be increasingly used to tackle the Visual Tracking problem. The first solution in this direction was the work of Yun *et al*. [52], which proposed an Action-Decision network to learn a policy for selecting a discrete number of actions to modify iteratively the bounding box in the previous frame. Huang *et al*. [17] used a Deep-Q-Network [34] to learn a policy for adaptively selecting efficient image features during the tracking process. In the work of [19], the tracker was modeled as an agent that takes decisions during tracking whether: to continue tracking with a state-of-the-art tracker or to re-initialize it; and to update or not the appearance model of the target object. In [4], authors used a variant of REINFORCE [49] to develop a template selection strategy, encouraging the tracking agent to choose, at every frame, the best template from a finite pool of candidate templates. In [39], authors presented a tracker which, at every time step, decides to shift the current bounding box while remaining on the same frame, to stop the shift process and move to the next frame, to update on-line the weights of the model or to re-initialize the tracker if the target is considered lost. Finally, [3] proposed to substitute the discrete action framework of [52] with continuous actions, thus performing just a single action at every frame.

All the presented methods include a pre-training step that uses SL to build a baseline policy or some other module used later by the tracking agents. Only after, RL is used to fine-tune such policies and modules. We take inspiration from RL methods that exploit expert demonstrations and we propose a novel end-to-end methodology based on on-policy Actor-Critic framework [32] to train a DNN capable of tracking generic objects in videos. We also demonstrate that the state value function learned during training, can be directly used to exploit the expert during tracking, in order to adjust wrong tracking behaviors and to consequentially improve the tracking accuracy.

3. Methodology

The key idea of this paper is to take advantage of an expert tracker for training and tracking. RL and expert demonstrations are used to train a DNN which is then capable of tracking autonomously a generic target object in a video. The same network is also capable of evaluating its own performance and the one of the expert, thus exploiting the latter's knowledge in potential failure cases.

3.1. Problem setting

In our setting, the tracking problem follows the definition of a Markov Decision Process (MDP). The tracker is treated as an artificial agent which interacts with an environment that is obtained as an MDP defined over a video. MDPs are a standard formulation for RL tasks and are composed of: a set of states $\mathcal{S}$; a set of actions $\mathcal{A}$; a state transition function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$; a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and a discount value $\gamma \in \mathbb{R}$.

The interaction with the video, which we call an episode, happens through a temporal sequence of observations $s_1, s_2, \cdots, s_t$, actions $a_1, a_2, \cdots, a_t$ and rewards $r_1, r_2, \cdots, r_t$. In the t-th frame, the agent is provided with the state s_t and outputs the continuous action a_t which consists in the relative motion of the target object, i.e. it indicates how its bounding box, which is known in frame $t-1$, should move to enclose the target in the frame t. This approach is similar to the MDP formulation given by Chen et al. [3], however we propose different definitions for the states, actions and rewards.

Preliminaries. Given a dataset $\mathcal{D} = \{\mathcal{V}_0, \cdots, \mathcal{V}_{|\mathcal{D}|}\}$, we consider the j-th video

$$\mathcal{V}_j = \left\{F_t \in \{0, \cdots, 255\}^{w \times h \times 3}\right\}_{t=0}^{T_j} \tag{1}$$

as a sequence of frames F_t. Let $b_t = [x_t, y_t, w_t, h_t]$ be the t-th bounding box defining the coordinates of the top left corner, and the width and height of the rectangle that contains the target object. At time $t-1$, given F_{t-1} and b_{t-1}, the goal of the tracker is to predict the bounding box b_t that best fits the target in the consecutive frame F_t.

State. Every state $s_t \in \mathcal{S}$ is defined as a pair of image patches obtained by cropping frames F_{t-1} and F_t using the bounding box b_{t-1}. Specifically, $s_t = \rho(F_{t-1}, F_t, b_{t-1}, k)$, where $\rho(\cdot)$ crops the frames F_{t-1}, F_t within the area of the bounding box $b'_{t-1} = [x'_{t-1}, y'_{t-1}, k \cdot w_{t-1}, k \cdot h_{t-1}]$ that has the same center coordinates of b_{t-1} but which width and height are scaled by k. With this function and by choosing $k > 1$, we can control the amount of additional image context information that is provided to the agent.

Actions and State Transition. Each action $a_t \in \mathcal{A}$ consists in a vector $a_t = [\Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t] \in [-1, 1]^4$ which defines the relative horizontal and vertical translations ($\Delta x_t, \Delta y_t$, respectively) and width and height scale variations ($\Delta w_t, \Delta h_t$, respectively) that have to be applied to b_{t-1} to predict the bounding box b_t. This is obtained through $\psi : \mathcal{A} \times \mathbb{R}^4 \to \mathbb{R}^4$ such that

$$\psi(a_t, b_{t-1}) = \begin{cases} x_t = x_{t-1} + \Delta x_t \cdot w_{t-1} \\ y_t = y_{t-1} + \Delta y_t \cdot h_{t-1} \\ w_t = w_{t-1} + \Delta w_t \cdot w_{t-1} \\ h_t = h_{t-1} + \Delta h_t \cdot h_{t-1} \end{cases} \tag{2}$$

After performing the action a_t, the agent moves from the state s_t into the state s_{t+1} which is defined as the pair of cropped images obtained from the frames F_t and F_{t+1} using the bounding box b_t.

Reward. The reward function $r(s_t, a_t)$ expresses the quality of the action a_t taken at state s_t. Our reward definition is based on the Intersection-over-Union (IoU) metric computed between b_t and the ground-truth bounding box, denoted as g_t, *i.e.*, $\text{IoU}(b_t, g_t) = (b_t \cap g_t)/(b_t \cup g_t) \in [0, 1]$. At every interaction step t, the reward is formally defined as

$$r(s_t, a_t) = \begin{cases} \omega\left(\text{IoU}(b_t, g_t)\right) \text{ if } \text{IoU}(b_t, g_t) \geq 0.5 \\ -1 \text{ otherwise} \end{cases} \tag{3}$$

with

$$\omega(z) = 2(\lfloor z \rceil_{0.05}) - 1 \tag{4}$$

flooring to the closest 0.05 digit, then shifting the input range from $[0, 1]$ to $[-1, 1]$.

Expert demonstrations. To guide the learning of our tracking agent we take advantage of the positive demonstrations of an expert tracker. Given $\mathcal{V}_j$, the bounding box prediction of the expert at time t is denoted as $b_t^{(d)}$. The demonstrations are obtained as sequences of triplets $\{(s_t^{(d)}, a_t^{(d)}, r_t^{(d)})\}_{t=0}^{T_j}$, each containing a state, an action and a reward, respectively. Precisely, we have that $s_t^{(d)} = \rho(F_{t-1}, F_t, b_{t-1}^{(d)}, k)$ and $a_t^{(d)} =$

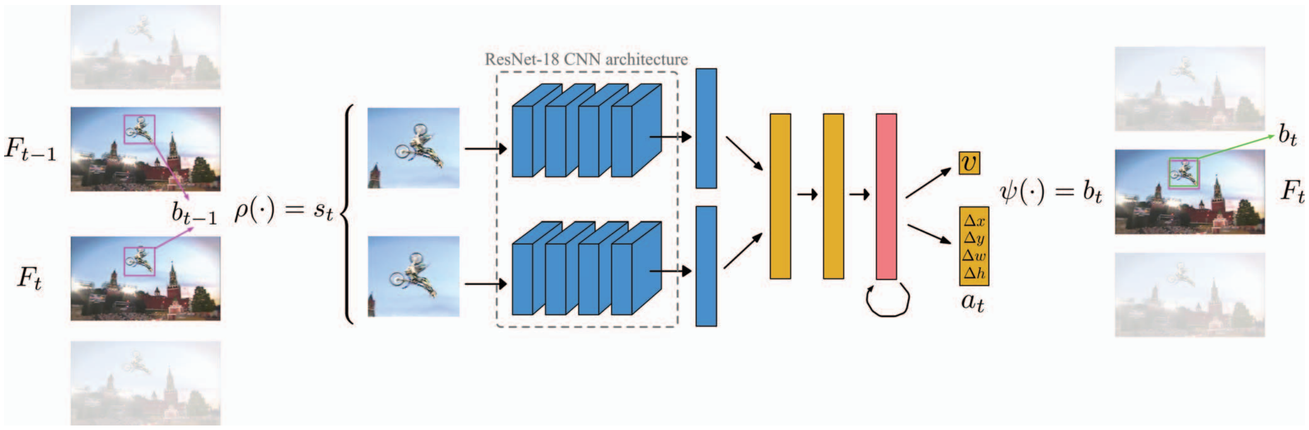

Figure 1. Visual representation of the interaction between the tracking agent and a video. Each pair of frames F_{t-1}, F_t is cropped by the function $\rho(\cdot)$ using the bounding box b_{t-1}. The obtained state s_t is fed to the agent's DNN which is composed by two branches of convolutional layers (the blue boxes) followed by, two fully-connected layers (rectangles in yellow), an LSTM layer (in light red) and two other fully connected layers for the prediction of v and the action a_t. Finally, the output bounding box b_t is built by the function $\psi(\cdot)$ which moves b_{t-1} by the relative shift a_t.

$[\Delta x_t^{(d)}, \Delta y_t^{(d)}, \Delta w_t^{(d)}, \Delta h_t^{(d)}]$, where its elements are obtained through $\phi : \mathbb{R}^4 \times \mathbb{R}^4 \rightarrow \; A$, defined as

$$\phi(b_t^{(d)}, b_{t-1}^{(d)}) = \begin{cases} \Delta x_t^{(d)} = (x_t^{(d)} - x_{t-1}^{(d)})/w_{t-1}^{(d)} \\ \Delta y_t^{(d)} = (y_t^{(d)} - y_{t-1}^{(d)})/h_{t-1}^{(d)} \\ \Delta w_t^{(d)} = (w_t^{(d)} - w_{t-1}^{(d)})/w_{t-1}^{(d)} \\ \Delta h_t^{(d)} = (h_t^{(d)} - h_{t-1}^{(d)})/h_{t-1}^{(d)} \end{cases} \tag{5}$$

Rewards are calculated as $r_t^{(d)} = r(s_t^{(d)}, a_t^{(d)})$.

3.2. Agent architecture

Our tracking agent maintains representations of both the policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ and the state value function $v : \mathcal{S} \rightarrow \mathbb{R}$. This is done by using a DNN with parameters θ. In particular, we used a deep architecture that is similar to the one proposed by Gordon *et al*. [10].

The network gets as input two image patches. These pass through two convolutional branches that have the form of ResNet-18 CNN architecture [12] and which weights are pre-trained for image classification on the ImageNet dataset [8]. The two tensors of feature maps produced by the branches are first linearized, then concatenated together and finally fed to two consecutive fully connected layers with ReLU activations. After that, the features are inputted to an LSTM [16] RNN. Both the fully connected layers and the LSTM are composed of 512 neurons. The output of the LSTM is finally fed to two separate fully connected heads, one that outputs the action $a_t = \pi(s_t|\theta)$ and the other that outputs the value of the state, i.e. $v(s_t|\theta)$.

In Figure 1 a visual representation of the DNN architecture, together with the interaction process, is presented.

3.3. Training

The proposed DNN is trained solely off-line and in an end-to-end manner. The implemented training procedure is based on the on-policy A3C [32] RL framework. This method exploits P parallel and independent agents that interact with their own environments and that later use the gained experience to update asynchronously the weights θ which are shared among all agents. Indeed, each agent owns a copy θ' of the weights and this is synchronized with θ after every learning step. A3C is a standard algorithm in RL, however it is not designed to take expert demonstrations into account. To overcome this limitation for our problem, we set up an A3C framework where a first half of the learning agents performs the traditional A3C learning, while the other half learns to imitate the actions of the expert tracker demonstrator in a supervised fashion.

Imitating agents. Each imitating agent interacts with its environment by observing states, performing actions and receiving rewards just as standard A3C agents. Every t_{max} steps the agent updates the weights θ of the shared model with the gradients of the following loss function

$$\mathcal{L}_{imit} = \sum_{i=1}^{t_{max}} |\phi(b_t^{(d)}, b_{t-1}) - a_t| \cdot m_i. \tag{6}$$

which is the L1 loss between the actions performed by the learning agent and the actions that the expert tracker would take to move the agent's bounding box b_{t-1} into the expert's $b_t^{(d)}$. These absolute values are masked by the values $m_i \in \{0, 1\}$. Each of these is computed during the interaction and determines the situation in which the agent performed worse than demonstrator ($m_i = 1$) or better ($m_i = 0$). By optimizing the loss function 6, the weights θ are changed only if the agent's performance, in terms of received reward, is lower than the performance of the expert tracker. In simple words, the demonstrator is used to learn a baseline behavior on which the RL agent can build up its own tracking strategy, thus reducing the random exploration and consequentially speed up the learning process.

RL agents. The training process performed by RL agents follows the standard structure proposed by Mnih et al. [32] for continuous control. Each agent interacts with the environment for a maximum of t_{max} steps. However, differently from the imitating agents, at each step t the RL agents sample actions from a normal distribution $\mathcal{N}(\mu, \sigma)$, where the mean is the predicted action, $\mu = \pi(s_t|\theta')$, and the standard deviation is obtained as $\sigma = |\pi(s_t) - \phi(g_t, b_{t-1})|$ (which is the absolute value of the difference between the agent's action and the action that obtains, by shifting b_{t-1}, the ground-truth bounding box g_t). Intuitively, σ shrinks $\mathcal{N}$ when the action a_t is close to the ground-truth action $\phi(g_t, b_{t-1})$, thus reducing the chance of choosing potential wrong actions when approaching the correct one. On the other hand, when the action a_t is far from $\phi(g_t, b_{t-1})$, σ takes a greater value, spreading $\mathcal{N}$. This allows the agent to explore more the environment and discover potential good actions.

Curriculum strategy. In addition to the guiding process done by the imitating learners using the expert demonstrations, we designed a curriculum learning strategy [1] to further facilitate the training. In a similar way as proposed by [40], we built a curriculum based on the performance of the learning agents w.r.t. to the expert demonstrator. In particular, after terminating each episode, a success counter is increased if the agent performs better than the expert in that episode, i.e. if the former's cumulative reward, received up to $\widehat{T}_j$, is greater or equal to the one obtained by the latter. In formal terms, the success counter is updated if the following holds

$$\sum_{i=1}^{\widehat{T}_j} r_i \geq \sum_{i=1}^{\widehat{T}_j} r_i^{(d)}. \tag{7}$$

The counter update is done by testing agents that interact with the sequences by performing $\pi(s_t|\theta')$ using a local copy θ' of the shared weights. The terminal episode index $\widehat{T}_j$ is successively increased during the training procedure by a central process which checks if the ratio, between the number of episodes in which the learning agent performs better than the demonstrator and the total number of episodes terminated, is above the threshold τ. With this learning setting, we ensure that at every augmentation of $\widehat{T}_j$ the agents face a simpler learning problem where they are likely to succeed and in a shorter time, since they have already developed a tracking policy that, up to $\widehat{T}_j - 1$, is at least good as the one of the expert.

3.4. Tracking at test time

Despite the fact that the proposed tracker is trained by taking advantage of an expert's knowledge, our tracker develops a tracking ability that can be exploited independently from the tracking strategy used by the demonstrator. Nevertheless, it is possible to take advantage of the expert's tracking performance also during the tracking phase. Therefore we set up two tracking strategies, the first one that tries to track autonomously the target object and we refer it as A3CT, and the second one that takes advantage of the demonstrator's knowledge also during tracking and that we name A3CTD.

A3CT. In this setting, A3CT is applied straight away on an arbitrary sequence. Each tracking sequence $\mathcal{V}_j$, with target object outlined by g_0, is considered as the MDP described in section 3.1. The tracker computes states s_t from frames F_t, performs actions as by means of the learned policy $a_t = \pi(s_t|\theta)$ which are used to output the bounding boxes $b_t = \phi(a_t, b_{t-1})$. At the beginning, $b_0 := g_0$.

No online update of the network's weights nor of the LSTM's hidden state are performed.

A3CTD. During training, the tracking agent learns both the policy $\pi(s_t|\theta)$ and the value function $v(s_t|\theta)$. $v(\cdot)$ is a function that predicts the reward that the agent expects to receive from the current state s_t to the end of the sequence. Since our reward definition is a direct measure of the IoU between the predictions of the agent and the ground-truth bounding boxes, $v(s_t|\theta)$ gives an estimate of the total amount of IoU that the tracker expects to obtain from state s_t on wards. This function can be exploited as a performance evaluation for both our tracker and the expert demonstrator. In particular, at each time step t, $\widehat{R} = v(s_t|\theta)$ and $\widehat{R}^{(d)} = v(s_t^{(d)}|\theta)$ are obtained as the evaluation for A3CTD and the expert tracker respectively. The expert state $s_t^{(d)}$ is obtained by cropping frames F_{t-1}, F_t using its previous prediction $b_{t-1}^{(d)}$. By comparing $\widehat{R}$ and $\widehat{R}^{(d)}$, our strategy decides if to output the bounding box of A3CTD or the bounding box produced by the expert tracker. More formally, if $\widehat{R} \geq \widehat{R}^{(d)}$ then the tracker outputs $b_t := \phi(a_t, b_{t-1})$ otherwise it outputs $b_t := b_t^{(d)}$.

3.5. Implementation details

In this section we report the results of the hyperparameters search which led to the best performance.

Before being fed to the DNN, the image crops that forms the MDP states are resized to $[128 \times 128 \times 3]$ pixels and standardized, per channel, by subtracting the mean and dividing by the standard deviation calculated on the ImageNet dataset [8]. The dilating factor k is set to 1.5.

A total number of $P = 16$ training agents was used. The discount factor γ was set to 1. The length of the rollout was defined in $t_{max} = 5$ steps. τ was set to 0.25. The model was trained for 40000 episodes using the Adam optimizer [21]. The learning rate for both imitating and training agents was set to 10^{-6}. A weight decay of 10^{-4} was also added to the L1 loss of the imitating agents as regulatory term.

Training and experiments have been conducted running our Python code with the PyTorch [38] machine learning library on an Intel Xeon E5-2690 v4 @ 2.60GHz CPU with 320 GB of RAM, four NVIDIA TITAN V GPUs and an NVIDIA TITAN Xp GPU each with 12 GB of memory. The training took around 4 days. In the evaluation of trackers' speed, we ignore disk read times since they do not dependent on the tracking algorithm.

Expert Tracker. The role of expert tracker was assigned to SiamFC [2]. The choice was motivated by the fact that this solution is nowadays an established methodology in the visual tracking panorama, and it shows great balance in results across many different benchmarks. In particular, SiamFC has currently one of the best performance on the public leader-board of the GOT-10k test set. Additionally, the source code was publicly available.

To obtain tracking demonstrations, we ran SiamFC on the training set of GOT-10k dataset [18]. The implemented SiamFC was trained on the ImageNet VID dataset [8]. This is an important aspect because, to train our tracking agent, we want examples of the tracker's real behaviour, that must be obtained on never seen before sequences. Moreover, demonstrations that are clearly useful are needed. So, of all the trajectories produced, we retained just the ones considered positive, i.e. the trajectories that satisfy $\text{IoU}(b_t^{(d)}, g_t) > 0.5$ for all $t \in \{1, \ldots, T_j\}$. All the others were discarded.

Training Dataset. To train A3CT and A3CTD we leveraged of the training set of the GOT-10k dataset [18]. This is a large-scale dataset containing 9335 training videos, 180 validation videos and other 180 videos for testing. In total, this dataset provides 1.5M bounding boxes that identify 10k different target objects. The latters belong to 563 distinct object classes. The actual number of training sequences we used is however inferior. In fact, just the videos which obtained a positive demonstration from the expert tracker were employed for training. Furthermore, as we aimed to take part to the VOT 2019 challenge, we removed 1000 sequences from the training set. These overlapped with the pool of videos used by the VOT committee for evaluation. After these pruning steps, the total amount of training samples, $|\mathcal{D}|$, resulted in 1782 videos.

4. Experiments

In this section we report the experimental setup and we discuss the results, obtained by the proposed trackers A3CT and A3CTD, on the benchmarks GOT-10k [18], LaSOT [9], UAV123 [35], OTB-100 [50], VOT-2018 [26] and VOT-2019.

4.1. GOT-10k Test Set

The GOT-10k [18] test set comprises 180 videos. Target objects belong to 84 different classes and 32 forms of object motion are present. To ensure a fair evaluation, the trackers that are evaluated on this benchmark are forbidden from using external datasets for training. The evaluation protocol proposed by the authors is the one-pass evaluation (OPE) [50]. The metrics used are the average overlap (AO) and the success rates (SR) with overlap thresholds 0.5 and 0.75.

	KCF [14]	MDNet [37]	ECO [5]	CCOT [7]	GOTURN [13]	SiamFC [2]	SiamFCv2 [46]	ATOM [6]	**A3CT**	**A3CTD**
AO	0.203	0.299	0.316	0.325	0.347	0.348	0.374	0.556	**0.415**	**0.425**
$\text{SR}_{0.50}$	0.177	0.303	0.309	0.328	0.375	0.353	0.404	0.634	**0.477**	**0.495**
$\text{SR}_{0.75}$	0.065	0.099	0.111	0.107	0.124	0.098	0.144	0.402	**0.212**	**0.205**

Table 1. State-of-the-art comparison on the GOT-10k test set in terms of average overlap (AO), and success rates (SR) with overlap thresholds 0.5 and 0.75. Except for ATOM, both versions of our approach outperform the previous methods in all three measures.

In Table 1 we report the results of A3CT and A3CTD against the state-of-the-art. A3CT outperforms the state-of-the-art trackers which, at the time of writing, appear on the GOT-10k test set leaderboard. In particular, it has a better tracking performance w.r.t. to the demonstrator tracker SiamFC [2], with a performance gain of 6.7% and in AO, 12.4% in $\text{SR}_{0.50}$, and 11.4% in $\text{SR}_{0.75}$. A3CTD increases additionally the performance of A3CT, with an improvement of 1% in AO, 1.8 in $\text{SR}_{0.50}$ but with a loss of 0.7% in $\text{SR}_{0.75}$. We perform worse than ATOM [6], however we remark that these results are obtained considering just 1782 of the 9335 sequences (19%) contained in the GOT-10k training set.

4.2. OTB-100

The OTB-100 [50] benchmark is a set of 100 challenging videos and it is widely used in the tracking literature. The standard evaluation procedure for this dataset is the OPE method and the metrics used are the success plot and the precision plot. The Area Under the Curve (AUC) of these curves are referred as success score (SS) and precision scores (PS) respectively.

In Table 2 we report the success and and precision scores against state-of-the-art solutions. On this bench-

	GOTURN [13]	RE3 [10]	KCF [14]	SiamFC [2]	ACT [3]	MDNet [37]	ECO [5]	SiamRPN++ [30]	**A3CT**	**A3CTD**
SS	0.395	0.464	0.477	0.575	0.625	0.677	0.691	0.696	**0.419**	**0.535**
PS	0.534	0.582	0.693	0.762	0.859	0.909	0.910	0.914	**0.568**	**0.717**

Table 2. State-of-the-art comparison on the OTB-100 benchmark in terms of success score (SS) and precision score (PS).

mark, A3CT and A3CTD have lower performance than ECO [5], MDNet [37], SiamRPN++ [30] and the expert SiamFC [2]. However, A3CT still performs better than GOTURN [13]. A3CTD instead outperforms RE3 [10] and KCF [14], with a 5.8-7.1% performance gain in SS and 1.8-13.5% in PS. In this setting, the help of the expert tracker is crucial to improve the results of A3CT, which sees an improvement of 11.6% in SS and 14.9% in PS.

4.3. LaSOT

We performed evaluations of A3CT and A3CTD performance on the test set of LaSOT benchmark [9]. This dataset is composed of 280 videos with a total of more than 650k frames and an average sequence length of 2500 frames. To evaluate our tracker, we use the same methodology and metrics used for the OTB-100 experiments.

In Table 3 we present the results against state-of-the-art trackers. In this setting, in terms of SS A3CT performs comparably to ECO [5] and RE3 [10] but much better than GOTURN [13]. Also in this case, the aid of the expert tracker is crucial, which results in a increment of 10.9% in SS and of 12.2% in PS. A3CTD so outperforms the expert SiamFC [2] in SS by 7.9% and MDNet [37] by 1.8%. Both our trackers are however weaker than SiamRPN++ [30].

	KCF [14]	GOTURN [13]	ECO [5]	RE3 [10]	SiamFC [2]	MDNet [37]	SiamRPN++ [30]	**A3CT**	**A3CTD**
SS	0.178	0.214	0.324	0.325	0.336	0.397	0.496	**0.306**	**0.415**
PS	0.166	0.175	0.301	0.301	0.339	0.373	-	**0.246**	**0.368**

Table 3. State-of-the-art comparison on the LaSOT benchmark in terms of success score (SS) and precision score (PS).

4.4. UAV123

The UAV123 [35] is a benchmark composed of 123 videos acquired from low-altitude UAVs. The dataset is inherently different from traditional visual tracking benchmarks like OTB and VOT, since it offers sequences with an aerial point of view. To evaluate our trackers, we use the same methodology and metrics used for the OTB-100 experiments.

In Table 4 we present the scores against state-of-the-art trackers. A3CT performs 14%, 8.2% and 5.6% better, in terms of SS, than KCF [14], GOTURN [13] and ACT [3] respectively. A3CTD has a 9.4% SS and a 13.2% PS improvements than A3CT and these lead to outperform SiamFC [2], ECO [5] and MDNet [37] with a gain of, respectively, 4.2%, 4%, 3.7% in SS and 2.4%, 1.3%, 0.7% in PS.

	KCF [14]	GOTURN [13]	ACT [3]	RE3 [10]	SiamFC [2]	ECO [5]	MDNet [37]	SiamRPN++ [30]	**A3CT**	**A3CTD**
SS	0.331	0.389	0.415	0.514	0.523	0.525	0.528	0.613	**0.471**	**0.565**
PS	0.523	0.548	0.636	0.667	0.730	0.741	0.747	0.807	**0.622**	**0.754**

Table 4. State-of-the-art comparison on the UAV123 benchmark in terms of success score (SS) and precision score (PS).

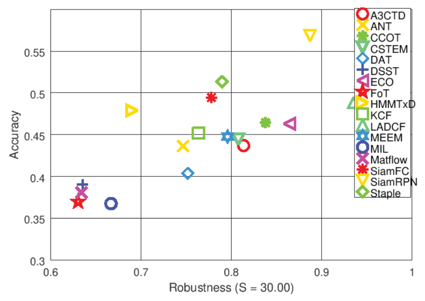

Figure 2. Accuracy-Robustness plot against some of the VOT-2018 [26] competitors.

4.5. VOT benchmarks

The VOT benchmarks are datasets used in the annual VOT tracking competition. These sets change year by year, introducing challenging tracking scenarios and increasing the difficulty of the task. Within the framework used by the VOT committee, trackers are evaluated based on Expected Average Overlap (EAO), Accuracy (A) and Robustness (R) [27]. We performed experiments on the test sets of VOT-2018 and VOT-2019 challenges. Both two benchmarks provide 60 (non completely overlapping) challenging videos.

VOT-2018. In Figure 2 we present the Accuracy-Robustness plot including A3CTD's performance in comparison with some of the partecipants to the VOT-2018 challenge. A3CTD achieves an EAO of 0.1847, an accuracy of 0.4536 while it failed (i.e. the IoU with the ground-truth becomes zero) 34.89 times. Our method perform definitely worse than the best solutions LADCF [51], SiamRPN [31] and ECO [5] that achieved an EAO of 0.3889, 0.3837 and 0.2809 respectively. A3CTD's performance is however comparable to the one of SiamFC [2], which achieved an EAO of 0.1875.

VOT-2019. At the time of writing, the results of VOT-2019 challenge are not available. We submitted just the A3CTD tracker, since it resulted in the best performance generally. It achieved an EAO of 0.1652 and of 0.1497 for the *baseline* and *realtime* experiments respectively. The overlap in the *baseline* experiment resulted in 0.4510.

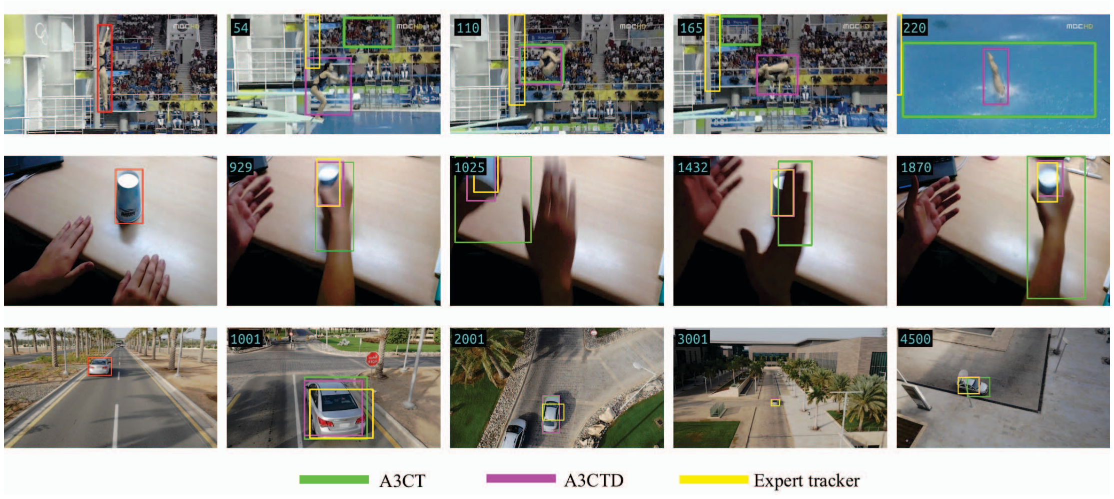

Figure 3. Qualitative examples of A3CT and A3CTD performance.

4.6. Ablation Study

To assess the validity of all the features of our proposed solution we performed an ablation study on the GOT-10k test set. In particular, we ran experiments where we trained A3CT and A3CTD without the curriculum strategy (A3CT-no-curr and A3CTD-no-curr respectively) and A3CT with just imitating agents (A3CT-SL). In Figure 4 we report the success plot with the comparison of the different models involved. A3CT-SL performs worse than A3CT, suggesting that the use of RL agents is crucial to improve the baseline behaviour learned by the imitating agents. Moreover, since the state value function is learned by RL agents, this setup does not allow to exploit the demonstrator in the tracking phase. A3CT-no-curr performs comparably to A3CT-SL, 4.1% lower than A3CT. The curriculum learning strategy allows the tracking agent to learn a more precise tracking policy. Interestingly, A3CTD-no-curr outperforms A3CTD by 2%. We believe that the increased length of the sequences during training allows the learning of a more accurate state value function, which is then able to make better predictions about the future behaviours of A3CT and the expert tracker. However, we chose A3CT and A3CTD as our final solution because of their lower difference in performance.

In terms of processing speed, A3CT runs at 90 FPS while A3CTD runs at 50 FPS.

Finally, in Figure 3 we present some qualitative examples of the tracking performance of A3CT and A3CTD.

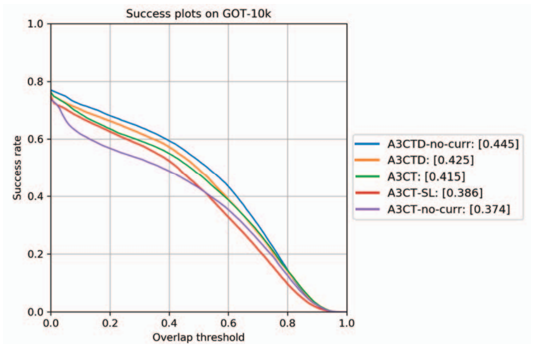

Figure 4. Success plot for the ablation study of A3CT and A3CTD.

5. Conclusions and Future Work

Thanks to the availability of a great amount of visual tracker and inspired by recent trends in RL, in this paper we proposed two novel trackers that are built on a deep regression network [13, 10]. The state-of-the-art tracking algorithm SiamFC [2] was executed on the large-scale tracking dataset GOT-10k [18] to obtain expert demonstrations. The proposed network was then trained inside an RL on-policy asynchronous Actor-Critic framework [32] that incorporated parallel SL agents. Experiments showed that the proposed A3CT and A3CTD trackers outperform state-of-the-art methods on the most recent the GOT-10k test set [18], LaSOT [9], UAV123 [35] benchmarks, and perform comparably with the state-of-the-art on OTB-2015 [50] and VOT benchmarks [26]. Moreover, A3CT and A3CTD achieved a processing speed of 90 and 50 FPS respectively and thus are suitable for real-time applications.

Future works will focus on the integration of more expert trackers. In particular, we will study how the performance of our proposed trackers change when different expert trackers and when pools of experts are considered as demonstrators.

References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, New York, New York, USA, 2009. ACM Press.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-Convolutional Siamese Networks for Object Tracking. jun 2016.

[3] B. Chen, D. Wang, P. Li, S. Wang, and H. Lu. Real-time 'Actor-Critic' Tracking. In *The European Conference on Computer Vision (ECCV)*, 2018.

[4] J. Choi, J. Kwon, and K. M. Lee. Visual Tracking by Reinforced Decision Making. *CoRR*, abs/1702.0, 2017.

[5] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ECO: Efficient Convolution Operators for Tracking. In *International Conference on Computer Vision and Pattern Recgnition*, nov 2017.

[6] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ATOM: Accurate Tracking by Overlap Maximization. In *International Conference on Computer Vision and Pattern Recognition*, nov 2018.

[7] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking. In *European Conference on Computer Vision*, aug 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, jun 2009.

[9] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. LaSOT: A High-quality Benchmark for Large-scale Single Object Tracking. In *International Conference on Computer Vision and Pattern Recognition*, sep 2019.

[10] D. Gordon, A. Farhadi, and D. Fox. Re3: Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects. *IEEE Robotics and Automation Letters*, 3(2):788–795, 2018.

[11] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang. Learning Dynamic Siamese Network for Visual Object Tracking. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1781–1789. IEEE, oct 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *International Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] D. Held, S. Thrun, and S. Savarese. Learning to Track at 100 {FPS} with Deep Regression Networks. *European Conference on Computer Vision*, abs/1604.0, 2016.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *CoRR*, abs/1404.7, 2014.

[15] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys. Deep Q-learning from Demonstrations. In *AAAI*, apr 2018.

[16] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[17] C. Huang, S. Lucey, and D. Ramanan. Learning Policies for Adaptive Tracking with Deep Feature Cascades. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 105–114. IEEE, oct 2017.

[18] L. Huang, X. Zhao, and K. Huang. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. oct 2018.

[19] J. S. S. III and D. Ramanan. Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning. *CoRR*, abs/1707.0, 2017.

[20] B. Kang, Z. Jie, and J. Feng. Policy Optimization with Demonstrations. In *ICML 2018*, volume 80, pages 2474–2483, 2018.

[21] D. P. Kingma and J. Ba. Adam: {A} Method for Stochastic Optimization. *CoRR*, abs/1412.6, 2014.

[22] V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, 2000.

[23] J. . L. A. . F. M. . C. L. . F. G. . V. T. . H. G. . N. G. . P. Kristan, Matej ; Matas.

[24] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, G. Häger, A. Lukežič, G. Fernández, et al. The Visual Object Tracking VOT2016 Challenge Results. In *European Conference on Computer Vision*, pages 777–823. Springer, Cham, 2016.

[25] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. C. Zajc, T. Vojir, et al. The Visual Object Tracking VOT2017 Challenge Results. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1949–1972. IEEE, oct 2017.

[26] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Č. Zajc, T. Vojír, G. Bhat, A. Lukežič, A. Eldesokey, G. Fernández, et al. The Sixth Visual Object Tracking VOT2018 Challenge Results. In *European Conference on Computer Vision*, pages 3–53. Springer, Cham, sep 2019.

[27] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A Novel Performance Evaluation Methodology for Single-Target Trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, nov 2016.

[28] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojí, G. Fernández, A. Lukežič, et al. The Visual Object Tracking VOT2014 Challenge Results. In *European Conference on Computer Vision*, pages 191–217. Springer, Cham, 2015.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[30] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. dec 2018.

[31] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High Performance Visual Tracking with Siamese Region Proposal Network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8971–8980. IEEE, jun 2018.

[32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, abs/1602.0, 2016.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5, 2013.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[35] M. Mueller, N. Smith, and B. Ghanem. A Benchmark and Simulator for UAV Tracking. In *European Conference on Computer Vision*, pages 445–461. Springer, Cham, 2016.

[36] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 6292–6299. Institute of Electrical and Electronics Engineers Inc., sep 2018.

[37] H. Nam and B. Han. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. *CoRR*, abs/1510.0, 2015.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[39] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou. Deep Reinforcement Learning with Iterative Shift for Visual Tracking. In *The European Conference on Computer Vision (ECCV)*, 2018.

[40] T. Salimans and R. Chen. Learning Montezuma's Revenge from a Single Demonstration. In *Proceedings of NeurIPS 2018 Workshop on Deep RL*, dec 2018.

[41] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the Game of {Go} with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, 2016.

[42] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354—-, 2017.

[43] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual Tracking: An Experimental Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[44] Y. Song, C. Ma, L. Gong, J. Zhang, R. Lau, and M.-H. Yang. CREST: Convolutional Residual Learning for Visual Tracking. In *Internationaò Conference on Computer Vision*, aug 2017.

[45] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

[46] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-End Representation Learning for Correlation Filter Based Tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5000–5008. IEEE, jul 2017.

[47] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. jul 2017.

[48] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[49] R. J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256, may 1992.

[50] Y. Wu, J. Lim, and M.-H. Yang. Online Object Tracking: A Benchmark. In *CVPR*, pages 2411–2418. IEEE Computer Society, 2013.

[51] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler. Learning Adaptive Discriminative Correlation Filters via Temporal Consistency Preserving Spatial Feature Selection for Robust Visual Tracking. *IEEE Transactions on Image Processing*, jul 2019.

[52] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi. Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1349–1358. IEEE, jul 2017.

[53] Z. Zhang and H. Peng. Deeper and Wider Siamese Networks for Real-Time Visual Tracking. In *International Conference on Computer Vision and Pattern Recognition*, jan 2019.

[54] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. Distractor-aware Siamese Networks for Visual Object Tracking. In *European Conference on Computer Vision*, aug 2018.