Real-time 'Actor-Critic' Tracking

Boyu Chen[1 [0000−0003−2397−7669]], Dong Wang[1* [0000−0002−6976−4004]], Peixia Li[1 [0000−0001−6167−5309]], Shuang Wang[2 [0000−0002−6462−6040]], and Huchuan Lu[1 [0000−0002−6668−9758]]

[1] School of Information and Communication Engineering,
Dalian University of Technology, China
[2] Alibaba Group, China
*Corresponding Author
bychen@mail.dlut.edu.cn, wdice@dlut.edu.cn,
pxli@mail.dlut.edu.cn, uu.ws@alibaba-inc.com,
lhchuan@dlut.edu.cn

Abstract. In this work, we propose a novel tracking algorithm with real-time performance based on the 'Actor-Critic' framework. This framework consists of two major components: 'Actor' and 'Critic'. The 'Actor' model aims to infer the optimal choice in a continuous action space, which directly makes the tracker move the bounding box to the object's location in the current frame. For offline training, the 'Critic' model is introduced to form a 'Actor-Critic' framework with reinforcement learning and outputs a Q-value to guide the learning process of both 'Actor' and 'Critic' deep networks. Then, we modify the original deep deterministic policy gradient algorithm to effectively train our 'Actor-Critic' model for the tracking task. For online tracking, the 'Actor' model provides a dynamic search strategy to locate the tracked object efficiently and the 'Critic' model acts as a verification module to make our tracker more robust. To the best of our knowledge, this work is the first attempt to exploit the continuous action and 'Actor-Critic' framework for visual tracking. Extensive experimental results on popular benchmarks demonstrate that the proposed tracker performs favorably against many state-of-the-art methods, with real-time performance.

Keywords: Visual tracking, Real-time tracking, Reinforcement learning

1 Introduction

Visual tracking aims to locate the target specified in the initial frame, which has many realistic applications such as video surveillance, augment reality and behavior analysis. In spite of many efforts having been done [1–3], it is still a challenge task due to many factors such as deformation, illumination change, rotation, occlusion, to name a few.

Deep-learning-based tracking algorithms have significantly improved the tracking performance in recent years [4–7]. The pre-trained convolutional neural networks (e.g., AlexNet, VGG-16 and VGG-M) are usually adopted to obtain rich feature representation for robust tracking. Among these methods, the MDNet tracker [5] achieves top-ranked performance in popular benchmarks (such as OTB-100 [8] and VOT2015 [9]).

(a) (b) (c)

Fig. 1. The search strategies of of different trackers. (a) MDNet [5]: search with random sampling; (b) ADNet [10]: iterative search with a series of discrete actions; and (c) our tracker: fast search with only one continuous action.

This method embeds the offline trained VGG-M network into the particle filter framework, where 256 candidate samples are randomly generated and each sample is verified using a CNN-based observation model in each frame. However, it is very slow due to the random sampling search strategy. To address this issue, Yun *et al.* [10] propose a reinforcement-learning-based tracker with an action-decision network (ADNet). This method takes a series of discrete actions to iteratively search the tracked object in each frame. Experimental results show that the ADNet tracker achieves slightly worse but three times faster performance than the MDNet method. The search strategies of MDNet and ADNet methods are illustrated in Figure 1 (a) and (b), respectively. We note that the learned iterative strategy in [10] is also far from the real-time requirement since it requires many iterative steps in every frame.

In this work, we develop a learning-based search strategy with continuous actions based on the 'Actor-Critic' framework. The core idea of this work is to predict only one continuous action using the 'Actor' model to locate the tracked object in each frame (see Figure 1 (c)). The reinforcement learning is exploited to offline train a good policy for determining the optimal action. In addition, the 'Critic' network acts as a verification scheme for both offline training and online tracking. The experimental results demonstrate that our 'Actor-Critic' tracker achieves better performance than other competing methods with real-time performance.

The contributions of this work can be summarized as follows.

(1) Our work is the first attempt to exploit the continuous actions for visual tracking. Visual tracking is treated as a dynamic search process where only one action output by the 'Actor' model is taken to locate the tracked object in each frame.

(2) Our work is also the first attempt to develop the 'Actor-Critic' tracking framework. The 'Actor-Critic' model is not only used to offline train the 'Actor' model based on reinforcement learning but also adopted to improve the robustness of our tracker during the tracking process. In addition, we improve the deep deterministic policy gra-

dient algorithm to effectively train our 'Actor-Critic' model with supervised learning and probability expert decision guidance.

(3) The proposed tracker is compared with some state-of-the-art trackers using the popular benchmarks, and the experimental results show that our tracker achieves good results with real-time performance.

2 Related Work

Visual Tracking. From the perspective of object localization, visual tracking can be treated as a dynamic search process to accurately locate the target in the current frame based on previous observations. Usually, this dynamic search process can be achieved with the sampling-verification framework. In each frame, a set of candidate states are randomly or densely sampled to describe possible object locations [11–13]. Then, an observation model is exploited to verify each candidate and determine the optimal state of the tracked object. However, the tracker with a robust observation model will be very slow since it requires calculating verification scores for a large number of sampled candidates, for both traditional methods [14–17] and deep visual trackers [5, 18, 19]. The correlation filter (CF) technique [20] could speed up verifying the densely sampled candidates with circulant matrix structures, resulting in many real-time trackers with good performance. Many attempts have been done to improve the original CF model in terms of feature combination [21, 22], scale estimation [23, 24], part-based extension [25, 26], multi-task learning [27, 28], bound effect [29, 30], to name a few. However, this speed merit of CF significantly whittled away when we combine it with deep features to pursuit higher accuracies (like HCFT [4], C-COT [31], ECO [7], LSART [32], DRT [33]).

Besides, the iterative search process can be adopted to conduct visual tracking, such as Meanshift [34], Lucas-Kanade [35] and their variants [36–39]. These methods are very efficient since they merely require a relative small of iterative steps (rather than a large number of sampled candidates) for locating the tracked object in each frame. However, their tracking accuracies are not satisfactory due to the following two reasons. First, the adopted low-level hand-crafted features limit their performance. Second, their search strategies are derived based on image or histogram gradients, without considering the high-level semantic information. Thus, the study of learning-based search strategies with deep neural networks may facilitate the trackers' taking a better trade-off between robustness and efficiency. Yun *et al.* [10] develop an ADNet tracking algorithm based on reinforcement learning, in which a series of iterative steps (corresponding to motion actions) are determined by the offline trained action-decision network. It speeds up the relevant MDNet method [5] more than three times without losing much accuracy. However, the learned iterative strategy in [10] is also far from the real-time requirement since it requires many iterative steps in each frame. In this work, we attempt to develop a learning-based search strategy with only one continuous action in each frame, which will significantly speed up the tracking method.

Reinforcement Learning. Reinforcement learning (RL) is a sequence learning method benefiting from trail and error, aiming to generate an agent for maximizing the cumulative future rewards. Due to the strong ability of deep neural networks, the RL technique has been applied on many computer vision tasks [40–42]. Recently, there exist some

attempts to exploit the RL technique for visual tracking. In [10], Yun *et al.* propose an action-decision network based on RL, which learns a good policy to select a series of actions from the action pool (including eleven candidate actions for translation moves, scale changes, and stop). Then, the tracker decides sequential actions to search the optimal position of the tracked object in the current frame, and then goes to deal with the next frame. In [43], Huang *et al.* exploit RL to learn an early decision policy for adaptively selecting efficient features during the tracking process. Based on the learned policy, eight discrete actions are taken to decide whether the tracker locates the tracked object on an early layer or continues processing subsequence layers. This method could effectively speed up the deep tracker without losing accuracy since it encourages the tracker to handle easy frames with cheap features while still to deal with difficult frames with expensive deep features. In [44], the tracker is modeled as an active agent to make online decisions whether the agent is still to 'track' or requires to 'reinitialize' and whether the current observation should be 'update' or 'ignored'. In [45], the RL method is utilized to construct a template selection strategy, encouraging the tracker choose the best template from finite candidate templates in every frame. Different from above-mentioned methods, we propose a novel 'Actor-Critic' tracking framework, which can effectively learn a good policy to determine one optimal action based on reinforcement learning.

3 Tracking via the 'Actor-Critic' Network

3.1 Overview

Visual tracking aims to infer the location of an arbitrary object in each subsequent frame given its initial position in the first frame. In this work, we attempt to conduct tracking within a novel 'Actor-Critic' framework, the overall pipeline of which is illustrated in Figure 2. The 'Actor' model aims to give one continuous action to directly make the tracker move the bounding box to the object location in the current frame. It can be effectively offline trained with a 'Critic' network based on deep reinforcement learning. During the tracking process, the 'Critic' model combines the action produced by 'Actor' to determine the quality of the action and facilitates improving the tracking performance. The details of our tracking framework are presented as follows.

3.2 Problem Settings

Considering tracking as a sequential decision problem, our algorithm follows Markov Decision Process (MDP). The basic components of MDP include states $s \in S$, actions $a \in A$, the state transition function $s^{'} = f(s, a)$, and the reward $r(s, a)$. In our MDP framework, the tracker is treated as an agent to infer the accurate bounding box of the tracked object in each frame. This agent is interacted with an environment through a sequence of observations $s_1, s_2, ..., s_t$, actions $a_1, a_2, ..., a_t$ and rewards $r_1, r_2, ..., r_t$. In the t-th frame, the agent gives the continuous action a_t according to the current state s_t and obtains the tracking result as $s_t^{'}$. In this work, the action a_t is defined as the relative motion of the tracked object indicating how its bounding box should move

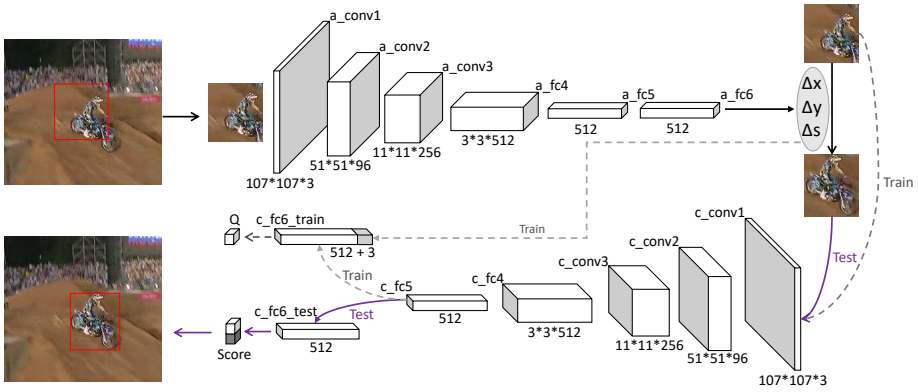


Fig. 2. The pipeline of the proposed tracking algorithm. 'a' means 'Actor' and 'c' means 'Critic'.

directly in frame t. Different with ADNet [10], our tracker takes only one continuous action to locate the tracked object, which makes our tracker more efficient. The detailed settings of s, a, $f(s,a)$ and r are presented as follows (we drop the frame index t for clarity in this subsection).

State. In this work, we define the state s as the observation image patch within the bounding box $b = [x, y, h, w]$, where (x, y) is the center position and h and w stand for the height and width respectively. To be specific, we define a per-processing function $s = \phi(b, F)$ to crop the image patch within the bounding box b in a given frame F and resize it to fit the input size of the deep network.

Action and State Transition. To conduct continuous control, the action space is assumed to be continuous, indicating how the bounding box should move directly. Here, we use the action $a = [\Delta x, \Delta y, \Delta s]$ to depict the relative motion of the tracked object, where Δx and Δy denote the relative horizontal and vertical translations and Δs stands for the relative scale change. Considering the temporal continuity in the tracking problem, we introduce some constraints to restrict the range of the action a: $-1 \leq \Delta x \leq 1$, $-1 \leq \Delta y \leq 1$ and $-0.05 \leq \Delta s \leq 0.05$. By applying the action a to the original bounding box b, the new bounding box $b^{'} = \left[x^{'}, y^{'}, h^{'}, w^{'}\right]$ can be obtained as

$$\begin{cases} x^{'} = x + \Delta x \times h \\ y^{'} = y + \Delta y \times w \\ h^{'} = h + \Delta s \times h \\ w^{'} = w + \Delta s \times w \end{cases} . \tag{1}$$

Then, the state transition process $s^{'} = f(s,a)$ can be implicitly achieved by applying the per-processing function $\phi\left(b^{'}, F\right)$.

In this work, we attempt to directly infer the optimal action a based on the state s using an 'Actor' model, i.e., $a = \mu(s|\theta^{\mu})$. $\mu(.)$ denotes the deep network for our 'Actor' model with the parameter θ^{μ}. The 'Actor' model is trained offline with the proposed

training strategy in Section 3.3, and then applied for online tracking. In practice, we exploit a double bounding box scheme to build the 'Actor' model (i.e., two bounding boxes of the target size and twice target size with the same center location).

Reward. The reward function $r(s, a)$ describes the localization accuracy improvement when transferring state s into state $s^{'}$ with a given action a. Thus, it can be defined based on the overlap ratio of the new bounding box $b^{'}$ and the ground truth G,

$$r(s,a) = \begin{cases} 1 & \text{if IoU}(b^{'}, G) > 0.7 \\ -1 & \text{else} \end{cases}, \tag{2}$$

where IoU denotes the Intersection-over-Union criterion (i.e., IoU(u, v) = $u \cap v / u \cup v$ for bounding boxes u and v). With every action, the reward will be generated and then be used to update the deep networks in offline learning.

3.3 Offline Training

Network Structure. Inspired by the recent successful trackers with lightweight deep networks, we use the pre-trained VGG-M model to initialize our 'Actor' and 'Critic' networks. As illustrated in Figure 2, there are three convolutional layers in both two networks, which are consistent with the first three convolutional layers of the VGG-M network. For the 'Actor' network, the next two fully connected layers with 512 output nodes are combined with the ReLU operation, and the last fully connected layer generates a three-dimensional output. For offline training, the 'Critic' model has similar structures with 'Actor' expect the last fully connected layer as it requires concatenating the three-dimensional action vector to obtain a Q-value for action evaluation according to the current state.

Training via DDPG. In this work, we train our 'Actor-Critic' network using the DDPG approach [46], the core idea of which is to iteratively update the 'Critic' and 'Actor' models with training sample pairs collected based on the RL rule. Given N pairs of $(s_i, a_i, r_i, s_i^{'})$, the 'Critic' model $Q(s, a)$ can be learned using the Bellman equation as in Q-learning. With the utilization of the target networks $\mu^{'}$ and $Q^{'}$, the learning process can be achieved by minimizing the following loss,

$$L = \frac{1}{N} \sum_{i} (y_i - Q(s_i, a_i | \theta^Q))^2, \tag{3}$$

where $y_i = r_i + \gamma Q^{'}(s_i^{'}, \mu^{'}(s_i^{'} | \theta^{\mu^{'}}) | \theta^{Q^{'}})$.

Then, the 'Actor' model can be updated by applying the chain rule to the expected return from the start distribution J with respect to the model parameters:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_{i} \nabla_a Q(s, a | \theta^Q)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s | \theta^\mu)\big|_{s=s_i}. \tag{4}$$

During the training iteration, we randomly select a piece of training sequences $[F_k, F_{k+1}, \ldots, F_{k+T}]$ with their ground truth $[G_k, G_{k+1}, \ldots, G_{k+T}]$ from the training data (k is the start frame number and T is the frame length). After that, we apply our tracker in the selected sequence to obtain a training pair $(s_t, a_t, r_t, s_t^{'})$ at frame t.

Algorithm 1 Offline training the 'Actor' network

Input: Training sequences [F] and their corresponding ground truths [G]
Output: Trained weights for the 'Actor' network

Initialize 'Critic' $Q(s,a)$ and 'Actor' $\mu(s|\theta^{\mu})$ with weights θ^{Q} and θ^{μ}.
Initialize the target network $Q^{'}$ and $\mu^{'}$ with weights $\theta^{Q^{'}} \leftarrow \theta^{Q}, \theta^{\mu^{'}} \leftarrow \theta^{\mu}$
Initialize the replay buffer R
repeat
Randomly select a piece of frames $[F_k, F_{k+1}, \ldots, F_{k+T}]$ with their ground truth $[G_k, G_{k+1}, \ldots, G_{k+T}]$
Receive initial observation state s_k according to F_k and G_k
Train the 'Actor' network for an iteration utilizing s_1
for each $t = 2, T+1$ **do**
1. Obtain state s_t according to state s_{t-1} and F_{k-1+t}
2. Select action $a_t = \mu(s_t|\theta^{\mu})$ according to the current policy and exploration probability ϵ;
3. Execute action a_t according to the transition in Eq. 1, observe reward r_t as Eq. 2 and the next state $s_t^{'}$
4. Store transition $(s_t, a_t, r_t, s_t^{'})$ in R
end for
Sample a random mini-batch of N transitions $(s_i, a_i, r_i, s_i^{'})$ from R
Update 'Critic' $Q(s,a)$ by minimizing the loss following Eq. 3
Update 'Actor' $\mu(s|\theta^{\mu})$ using the sampled policy gradient following Eq. 4
Update the target networks:

$$\begin{aligned} \theta^{Q^{'}} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q^{'}} \\ \theta^{\mu^{'}} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu^{'}} \end{aligned} \tag{5}$$

until Reward become stable

Training Process Improvement. It is not feasible to directly apply the original DDPG framework to train our model, since the action space is very huge in our tracking problem. Thus, we attempt to improve the training process from two following aspects.

(1) Due to the huge action space, it is difficult to obtain a positive reward when an agent follows a random exploration strategy for a given video clip. This will make the DDPG method less effective in training our model. To solve this problem, we utilize the supervised information from the first frame to initialize the 'Actor' model for adapting the current environment. That is, the 'Actor' model is fine-tuned through the adaptive moment estimation method to minimize the following L_2 loss function,

$$\min \frac{1}{M} \sum_{m=1}^{M} \left[\mu\left(s_m|\theta^{\mu}\right) - a_m\right]^2, \tag{6}$$

where M is the number of training samples and $\mu\left(.|\theta^{\mu}\right)$ denotes the 'Actor' network with parameter θ^{μ}. s_m is the m-th sampled state and a_m denotes its ground truth action.

(2) The initialization scheme above cannot fully solve the imbalance problem of positive and negative samples as there are many unpredicted challenges causing tracking drifts. Thus, we exploit expert decisions to guide the learning process. The original DDPG algorithm introduces an exploration policy $\mu^{'}$ by adding noise sampled from a noise process $\mathcal{N}$ to the actor policy $\mu(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$. However, this similar exploration mechanism is not suitable for our tracking task due to the huge action space. Therefore, we adopt a probabilistic expert decision guidance to supersede exploration mechanism in reinforcement learning. In a video sequence, with a certain probability ϵ, an expert decision guidance is applied to replace the action output by the 'Actor' network. The probability ϵ gradually decreases during the training process.

Our 'Actor' network can be effectively trained offline by the DDPG method with two improvements above. The overall training process is summarized in **Algorithm 1**.

3.4 Online Tracking

Network Initialization. To make our tracker further suitable for the current sequence, we initialize both 'Actor' and 'Critic' models with the ground truth in the first frame.

For 'Actor', we first sample M candidate bounding boxes $b_m|_{m=1}^{M}$ around the ground truth and calculate their corresponding accurate actions $a_m|_{m=1}^{M}$. Then, we extract the image observation s_m for the candidate location b_m using per-processing function $s_m = \phi\left(b_m, F\right)$ (defined in Section 3.2). Thus, the 'Actor' network can be fine-tuned using the Adam method to minimize the L2 loss $\frac{1}{M}\sum\limits_{m=1}^{M}\left[\mu\left(s_m|\theta^{\mu}\right) - a_m\right]^2$.

For online tracking, the 'Critic' model $v\left(s|\theta^{\nu}\right)$ is a classification network. To initialize it, we assign a binary label l_m to the m-th candidate using the following rule,

$$l_m = \begin{cases} 1 & \text{if IoU}(b_m, G) > 0.7 \\ 0 & \text{else} \end{cases}, \tag{7}$$

where G denotes the ground truth bounding box. With the collected image-label pairs $\{s_m, l_m\}|_{m=1}^{M}$, the 'Critic' network is initialized using the Adam method to minimize the following loss function,

$$\arg\min_{\theta^{\nu}} -\sum\nolimits_{s\in S_+} p_+(s|\nu;\theta^{\nu}) - \sum\nolimits_{s\in S_-} p_-(s|\nu;\theta^{\nu}), \tag{8}$$

where S_+ and S_- denote positive and negative training sets respectively. The 'Critic' network outputs the foreground and background probabilities $p_+(s|\nu;\theta^{\nu})$ and $p_-(s|\nu;\theta^{\nu})$ for a given state (or observation patch) s.

Tracking via 'Actor-Critic'. For online tracking, we exploit both 'Actor' and 'Critic' networks within a tracking-and-verification scheme. In the t-th frame, we first calculate the state s_t using the preprocessing function $\phi\left(b_{t-1}^{'}, F_t\right)$ ($b_{t-1}^{'}$ denotes the optimal object location in the $t-1$ frame and F is the image frame). Second, we put the state s_t into the the 'Actor' network resulting in the action a_t, i.e., $a_t = \mu\left(s_t|\theta^{\mu}\right)$. With the action a_t and location $b_{t-1}^{'}$, we can obtain the new location $b_t^{'}$ and its corresponding

state $s_t^{'}$ in the current frame. Then, we utilize the 'Critic' network to verify the observation $s_t^{'}$, i.e., $v\left(s_t^{'}|\theta^{\nu}\right)$. If the score given by the 'Critic' network is large than 0, we treat the action a_t being reliable and adopt the location $b_t^{'}$ as the optimal location in the t-th frame. Otherwise, we exploit a re-detection technique using the 'Critic' network to evaluate a series of sampled candidates $b_t^m|_{m=1}^M$ around $b_{t-1}^{'}$ (same as the sampling strategy in Network Initialization). After that, the optimal location $b_t^{'}$ is obtained as the candidate with the highest score outputed by 'Critic'.

Network Update. An effective updating strategy helps our tracker take a good trade-off between robustness and efficiency. The 'Actor' model has a stable performance during the tracking process due to our offline training, thus, we merely update the 'Critic' network when needed. If the verification score given by 'Critic' is less than 0, we think it does not fit well with the appearance change in the current environment and use positive and negative samples collected in previous 10 frames to update the network based on Eq.8.

3.5 Implementation Details

Samples generation. To train the networks in both offline and online tracking stages, we sample $X_t^i = \left(x_t^i, y_t^i, z_t^i\right), i = 1, \ldots, N$ (x and y are horizontal and vertical translations; z denotes the scale) from a Gaussian distribution centered by the object location in frame $t-1$. The covariance is a diagonal matrix $diag\left(0.09d^2, 0.09d^2, 0.25\right)$, where d is the mean of the width and height of the tracked object.

Offline Training. To train our 'Actor' network offline, we use 768 video sequences from the ImageNet Video [47] trainval set. We randomly choose continuous twenty to forty frames in a video for each iteration. For initializing the 'Actor' network in the first frame, we collect 32 samples whose IoU scores with ground truth are larger than 0.7. The learning rate is set to 1e-4 in the initialization stage.

The possibility of adopting the expert decision ϵ is set to 0.5 at first and reduced by 5% after every ten thousand iterations. We update the target networks every ten thousand iterations. τ in the target networks updating is set to 0.001. The learning rates of the 'Actor' and 'Critic' networks are set to 1e-6 and 1e-5, respectively. In addition, we use a replay buffer size of 10^4. We finish the training of the 'Actor' network after two hundred and fifty thousand iterations.

Online Tracking. For online tracking, we collect 500 positive samples and 5000 negative samples with ground truth in the first frame. Only positive samples are used for training the 'Actor' network. We initialize the 'Actor' network with learning rates 1e-4 until the loss is less than 1e-4 and initialize the 'Critic' network with learning rates 1e-4 for 30 iterations. The batch sizes for the 'Actor' and 'Critic' models are 64 and 128, respectively. When the predicted target location of the highest foreground score of all candidates are less than 0, we consider it as the tracking failure, and the re-detection is conducted to capture the missed target. We draw 256 samples for the re-detection scheme. Simultaneously, the 'Critic' model is online updated with collected samples from 10 recent successful tracking frames. We collect 50 positive samples and 150 negative samples from each successful tracking frame.

Fig. 3. Qualitative results of our ACT method and other trackers on some challenging sequences (ClifBar, Girl2, Matrix, MotorRolling, Skiing, Walking2).

4 Experiment

Our tracker is implemented in Python with the Pytorch framework, which runs at 30 fps on a PC with a 3.4GHz CPU with 32G memory and a TITAN GPU with 12G memory. The website of our ACT method is available on https://github.com/bychen515/ACT. Our tracker based on the 'Actor-Critic' network is denoted as ACT for clarity. We compare our tracker with many state-of-the-art trackers using standard tracking benchmarks such as Online Tracking Benchmark (OTB) [8, 48] and Visual Object Tracking challenge 2016 (VOT2016) [49]. Some representative visual results are shown in Figure 3.

4.1 Evaluation on OTB

In this subsection, we evaluate our tracker using both OTB-2013 [48] and OTB-2015 [8] datasets. The proposed tracker is compared with ten state-of-the-art trackers with real-

time performance including PTAV [50], CFNet [51], ACFN [52], SiameFC [6], ECO-HC [7], LCT [53], LMCF [54], Staple [22], DSST [55] and KCF [20]. The first four algorithms employ the feature descriptors from CNNs while the rest of the methods are based on the traditional hand-crafted features.

In this work, we adopt both precision and success plots to evaluate different trackers. The precision plot illustrates the percentage of frames where the center location error between the object location and ground truth is smaller than a per-defined threshold. Whereas the success plot demonstrates the percentage of frames the Intersection Over Union (IOU) of the predicted and the ground truth bounding boxes is higher than a given ratio. The trackers can be ranked the accuracy at 20 pixel threshold in the precision plot and the Area Under Curve (AUC) score in the success plot.

OTB-2013. We first evaluate our tracker in comparison with ten competing methods using the OTB-2013 dataset [48]. This dataset is one of the most popular benchmark including 50 fully-annotated video sequences with 11 various challenging factors such as fast motion, occlusion, illumination change, motion blur, and background clutter. These attributes could facilitate understanding the characteristics of our tracker.

Figure 4 (a) illustrates the precision and success plots over 50 sequences in OTB-2013. From this figure, we can see that our ACT method achieves best performance in terms of precision and the second best result in terms of success. These outstanding results are partly attributed to the strength of CNN features, which makes our tracker effectively depict the appearance of the tracked object compared with low-level hand-crafted features. In comparison with CFNet, SiameFC and ACFN methods using deep networks, our ACT algorithm still achieves better performance due to the proposed learning scheme for determining the accurate action. Table 1 summarizes the average precision scores of different trackers for 11 challenging attributes in OTB-2013. It can be seen from this table that our ACT method performs better in handling most of challenges. The ECO-HC and PTAV also achieve good performance due the improved correlation filter technique or the explicit combination of a tracker and a verifier.

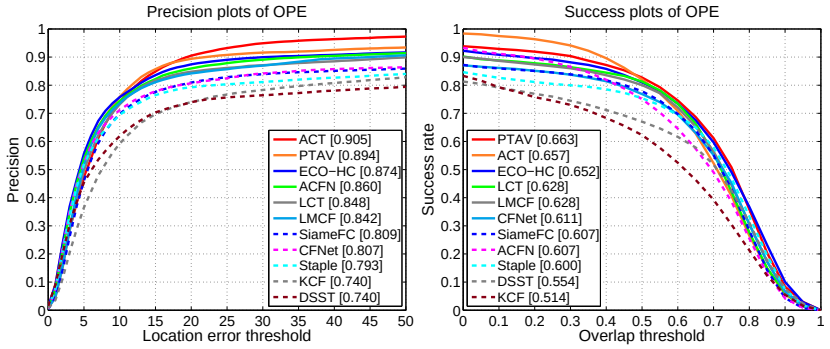
OTB-2015. Wu *et al.* [8] extend the OTB-2013 dataset with 50 more video, denoted as OTB-2015. The OTB-2015 dataset introduces more challenges for evaluating online tracking algorithms. Figure 4 (b) reports the precision and success plots of different trackers in OTB-2015. The results demonstrate that the proposed tracker is still very competitive compared with other methods.

4.2 Evaluation on VOT2016

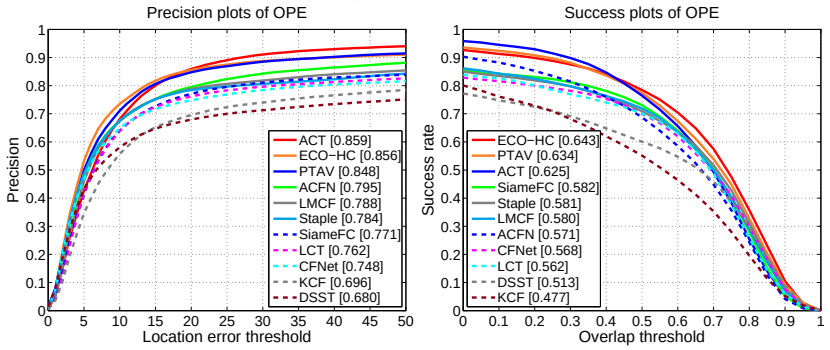
In addition, we report the evaluation results on the VOT2016 dataset [49], which contains 60 sequences with substantial variations.

Different from the OTB dataset, in the VOT challenge protocol, a tracker is re-initialized whenever tracking fails. The evaluation module reports both accuracy and robustness, corresponding to the total bounding box overlap ratio and the number of failures respectively. The VOT2016 challenge introduces the expected average overlap (EAO) as a new metric to rank tracking algorithms. It reflects the accuracy of the algorithm, with taking robustness into account.

Our algorithm is compared with seven trackers, which all join in the VOT2016 challenge. We report the average accuracy and robustness rank of all trackers in the



(a) OTB-2013



(b) OTB-2015

Fig. 4. The precision and success plots of different trackers on the OTB-2013 (a) and OTB-2015 (b) datasets. We can see that our ACT method performs better than other competing trackers.

Table 1. Average precision scores on different attributes: illumination variation (IV), out-of-plane rotation (OPR), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-view(OV), background cluttered (BC) and low resolution (LR). The best and the second best results are in red and blue colors, respectively.

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR	AV
ACT(Ours)	0.855	0.910	0.871	0.882	0.806	0.805	0.867	0.889	0.788	0.915	0.873	0.905
PTAV [50]	0.848	0.837	0.902	0.892	0.815	0.805	0.853	0.894	0.853	0.880	0.615	0.894
ECO-HC [7]	0.793	0.838	0.913	0.863	0.777	0.797	0.801	0.862	0.883	0.816	0.666	0.874
ACFN [52]	0.793	0.813	0.856	0.902	0.709	0.719	0.814	0.870	0.788	0.783	0.429	0.860
LCT [53]	0.792	0.758	0.845	0.873	0.664	0.665	0.802	0.850	0.728	0.796	0.352	0.848
LMCF [54]	0.783	0.775	0.844	0.869	0.714	0.730	0.779	0.826	0.695	0.848	0.555	0.842
SiameFC [6]	0.709	0.796	0.802	0.743	0.698	0.723	0.743	0.788	0.780	0.732	0.659	0.809
CFNet [51]	0.728	0.799	0.758	0.759	0.705	0.691	0.762	0.785	0.500	0.806	0.619	0.807
Staple [22]	0.741	0.733	0.787	0.812	0.688	0.643	0.773	0.773	0.679	0.753	0.550	0.793
DSST [55]	0.730	0.738	0.706	0.658	0.544	0.531	0.768	0.736	0.511	0.694	0.479	0.740
KCF [20]	0.728	0.679	0.749	0.740	0.650	0.602	0.725	0.729	0.650	0.753	0.381	0.740

Table 2. Besides, the EAO metric is also shown in this Table which gives the orders of all trackers. As illustrated in Table 2, our ACT method also achieves very competitive results. The C-COT and MLDF methods perform better than our ACT method, however, they merely run less than 2fps.

Table 2. The overall ranking score of accuracy (A), robustness (R) and expected overlap (EAO) in VOT2016.

Tracker	C-COT [31]	MLDF	ACT(Ours)	MDNet-N	SiamAN	SO-DLT [56]	KCF	DSST
Accuracy	1.87	2.77	2.25	1.63	2.37	2.23	2.98	2.60
Robustness	2.08	1.95	3.47	2.55	3.43	3.98	3.87	4.45
EAO	0.3310	0.3106	0.2746	0.2572	0.2352	0.2213	0.1924	0.1814

4.3 Analysis

Self-comparison. To verify the contribution of each component in our algorithm, we implement several variations of our approach and evaluate them using OTB-2013. These versions include: (1) 'ACT-vgg': the ACT method is not pertrained and simple adopts initial parameters of the VGG-M model to initialize the 'Actor' network; (2) 'ACT-rl': the ACT method without the reinforcement learning process; (3) 'ACT-init': the ACT method without using the initialization in the first frame among the training video sequence; (4) 'ACT-ex': the ACT method replacing the expert decision guidance by normal exploration in tradition DDPG methods. The performance of all variations and our final ACT is reported in Figure 5, from which we can see that all components facilitates improving the tracking performance. For examples, the comparison of the 'ACT-rl' and final ACT methods demonstrates the reinforcement learning process could effectively learn a good policy for action decision. The 'ACT-rl' method cannot learn the action policy from the training data. We note that the 'ACT-vgg' method without offline training runs at only 2fps with not good performance, which means the 'Actor' model without pre-trained cannot output the accurate action and the 'Critic' verification scheme requires to be invoked more frequently.

Compared with ADNet [10] and MDNet [5]. We note that the most relevant trackers of our ACT method are ADNet [10] and MDNet [5] since they adopt the VGG-M model as the basic network structure but with different search strategies. The detailed comparisons are reported in Table 3. The MDNet method performs the best in terms of accuracy but runs very slow due to the random sampling search strategy. The ADNet tracker exploits the iterative search strategy with few discrete actions in each frame, the fast version of which could achieve 15fps with losing about 3% accuracy compared with MDNet. Our ACT method performs slightly worse than the ADNet tracker and achieves comparable accuracies with the ADNet-fast one. However, our tracker runs at 30fps, twice than ADNet-fast and more than ten times than the original ADNet. This can be mainly attributed to the adopted continuous action, which locates the tracked object using merely one action in each frame.

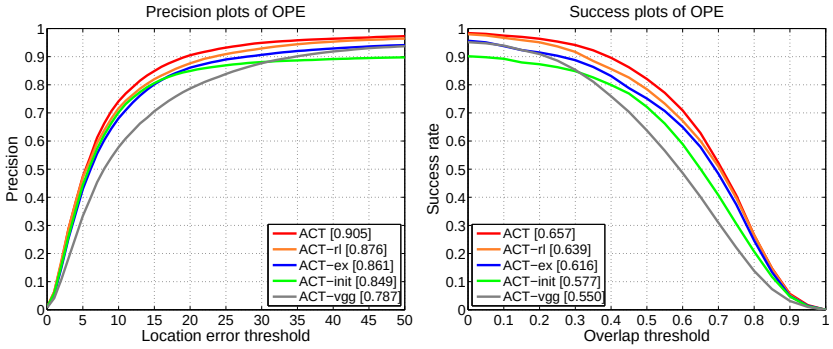


Fig. 5. Precision and success plots on OTB50 for different variations of our algorithm.

Table 3. The comparions of our ACT tracker with ADNet [10] and MDNet [5] methods in OTB.

Method	ACT(Ours)	ADNet [10]	ADNet-fast [10]	MDNet [5]
Prec.(20px) on OTB-2013	0.905	0.903	0.898	0.948
IOU(AUC) on OTB-2013	0.657	0.659	0.670	0.708
Prec.(20px) on OTB-2015	0.859	0.880	0.851	0.909
IOU(AUC) on OTB-2015	0.625	0.646	0.635	0.678
FPS	30	3	15	1

5 Conclusions

This work presents a novel 'Actor-Critic' tracking method based on reinforcement learning. The 'Actor' model acts as an action decision network to generate an optimal action that moves the bounding box to the object's location. Compared with existing algorithms, our method merely takes one continuous action in each frame, which makes it very efficient. For offline training, a 'Critic' network is integrated with the 'Actor' one to construct the 'Actor-Critic' framework, which can effectively learn the weights of the 'Actor' network. For online tracking, the similar 'Critic' network verifies the reliability of the output action and invokes the re-detection scheme if needed. Extensive experiments demonstrate that the proposed tracking algorithm achieves better performance than many state-of-the-art real-time trackers.

Acknowledgment. This paper was supported in part by the Natural Science Foundation of China #61751212, #61502070, #61725202, #61771088, #61472060, #61632006, #91538201, and in part by the Fundamental Research Funds for the Central Universities under Grant #DUT18JC30. This work was also supported by Alibaba Group through Alibaba Innovative Research (AIR) program.

References

1. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys **38**(4) (2006)

2. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A.R., van den Hengel, A.: A survey of appearance models in visual object tracking. ACM Transactions on Intelligent Systems and Technology **4**(4) (2013) 58:1–58:48
3. Li, P., Wang, D., Wang, L., Lu, H.: Deep visual tracking: Review and experimental comparison. Pattern Recognition **76** (2018) 323–338
4. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV. (2015)
5. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. (2016)
6. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV. (2016)
7. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ECO: efficient convolution operators for tracking. In: CVPR. (2017)
8. Wu, Y., Lim, J., Yang, M.: Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9) (2015) 1834–1848
9. Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Cehovin, L., Fernández, G., Vojír, T., Häger, G., Nebehay, G., Pflugfelder, R.P.: The visual object tracking VOT2015 challenge results. In: ICCV. (2015)
10. Yun, S., Choi, J., Yoo, Y., Yun, K., Choi, J.Y.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR. (2017)
11. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR. (2006)
12. Ross, D.A., Lim, J., Lin, R., Yang, M.: Incremental learning for robust visual tracking. International Journal of Computer Vision **77**(1-3) (2008) 125–141
13. Babenko, B., Yang, M., Belongie, S.J.: Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(8) (2011) 1619–1632
14. Jia, X., Lu, H., Yang, M.: Visual tracking via adaptive structural local sparse appearance model. In: CVPR. (2012)
15. Zhong, W., Lu, H., Yang, M.: Robust object tracking via sparse collaborative appearance model. IEEE Transactions on Image Processing **23**(5) (2014) 2356–2368
16. Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M., Hicks, S.L., Torr, P.H.S.: Struck: Structured output tracking with kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(10) (2016) 2096–2109
17. Li, Z., Zhang, J., Zhang, K., Li, Z.: Visual tracking with weighted adaptive local sparse appearance model via spatio-temporal context learning. IEEE Transactions on Image Processing **27**(9) (2018) 4478–4489
18. Wang, N., Yeung, D.: Learning a deep compact image representation for visual tracking. In: NIPS. (2013)
19. Li, H., Li, Y., Porikli, F.: DeepTrack: learning discriminative feature representations by convolutional neural networks for visual tracking. In: BMVC. (2014)
20. Henriques, J.F., Rui, C., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(3) (2015) 583–596
21. Zhu, G., Wang, J., Wu, Y., Zhang, X., Lu, H.: MC-HOG correlation tracking with saliency proposal. In: AAAI. (2016)
22. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: CVPR. (2016)
23. Li, Y., Zhu, J.: A scale adaptive kernel correlation filter tracker with feature integration. In: ECCVW. (2014)
24. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(8) (2017) 1561–1575

25. Li, Y., Zhu, J., Hoi, S.C.H.: Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In: CVPR. (2015)
26. Liu, T., Wang, G., Yang, Q.: Real-time part-based visual tracking via adaptive correlation filters. In: CVPR. (2015)
27. Tang, M., Feng, J.: Multi-kernel correlation filter for visual tracking. In: ICCV. (2015)
28. Zhang, T., Xu, C., Yang, M.: Multi-task correlation particle filter for robust object tracking. In: CVPR. (2017)
29. Danelljan, M., Hager, G., Khan, F.S., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV. (2015)
30. Galoogahi, H.K., Fagg, A., Lucey, S.: Learning background-aware correlation filters for visual tracking. In: ICCV. (2017)
31. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV. (2016)
32. Sun, C., Wang, D., Lu, H., Yang, M.H.: Learning spatial-aware regressions for visual tracking. In: CVPR. (2018)
33. Sun, C., Wang, D., Lu, H., Yang, M.H.: Correlation tracking via joint discrimination and reliability learning. In: CVPR. (2018)
34. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(5) (2003) 564–575
35. Baker, S., Matthews, I.A.: Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision **56**(3) (2004) 221–255
36. Jiang, N., Liu, W., Wu, Y.: Learning adaptive metric for robust visual tracking. IEEE Transactions on Image Processing **20**(8) (2011) 2288–2300
37. Li, P., Wang, Q.: Robust registration-based tracking by sparse representation with model update. In: ACCV. (2012)
38. Ning, J., Zhang, L., Zhang, D., Wu, C.: Robust mean-shift tracking with corrected background-weighted histogram. IET Computer Vision **6**(1) (2012) 62–69
39. Oron, S., Bar-Hillel, A., Avidan, S.: Extended lucas-kanade tracking. In: ECCV. (2014)
40. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: ICCV. (2015)
41. Bellver, M., Giro-i Nieto, X., Marques, F., Torres, J.: Hierarchical object detection with deep reinforcement learning. In: NIPS. (2016)
42. Jayaraman, D., Grauman, K.: Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In: ECCV. (2016)
43. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: ICCV. (2017)
44. III, J.R.S., Ramanan, D.: Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In: ICCV. (2017)
45. Choi, J., Kwon, J., Lee, K.M.: Visual tracking by reinforced decision making. CoRR **abs/1702.06291** (2017)
46. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. CoRR **abs/1509.02971** (2015)
47. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3) (2015) 211–252
48. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR. (2013)
49. Kristan, M., Leonardis, A., Jiri Matas, e.a.: The visual object tracking VOT2016 challenge results. In: ECCVW. (2016)
50. Fan, H., Ling, H.: Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In: ICCV. (2017)

51. Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: End-to-end representation learning for correlation filter based tracking. In: CVPR. (2017)
52. Choi, J., Chang, H.J., Yun, S., Fischer, T., Demiris, Y., Choi, J.Y.: Attentional correlation filter network for adaptive visual tracking. In: CVPR. (2017)
53. Ma, C., Yang, X., Zhang, C., Yang, M.: Long-term correlation tracking. In: CVPR. (2015)
54. Wang, M., Liu, Y., Huang, Z.: Large margin object tracking with circulant feature maps. In: CVPR. (2017)
55. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC. (2014)
56. Wang, N., Li, S., Gupta, A., Yeung, D.: Transferring rich feature hierarchies for robust visual tracking. CoRR **abs/1501.04587** (2015)