Actor-Critic Tracking with Precise Scale Estimation and Advantage Function

Chuyao Wang¹, a, † and Yuchen Ling², b, †

¹School of Mathematics, Computer Science & Engineering, City, University of London, London, UK
²School of Electronic Engineering, The Hong Kong University of Science and Technology, Hong Kong, China
aChuyao.wang@city.ac.uk, blyingad@connect.ust.hk
†These authors contributed equally.

Abstract. In this work, a deep reinforcement learning (DRL) method is proposed to address the problem of real-time object tracking. The adopted framework in this paper is based on the ‘Actor-Critic’ tracker (ACT), since ACT only considers the scale change instead of regression object boundary, which cannot adapt the object size variation. To this end, the ACT method is improved by using a more reasonable action space, which contains a left-top and right-bottom corner coordinates. Precise shape estimation is given by regressing the variation of width and height, respectively. Furthermore, to speed up the whole training and tracking process, the Advantage Function (AF) is adopted, and its performance is compared with ACT, ACT with improved action space (IAS), and ACT with IAS and AF. This method is tested on the OTB100 dataset to validate its effectiveness.

1. Introduction

Visual tracking has been a popular topic in the artificial intelligence field for its application on many occasions, such as automatic driving, and video surveillance. However, it is still a hard problem because many factors and complicated dynamic environments should be considered in the algorithm. Researchers should also consider many aspects of the tracker’s performance.

In recent years, there appear many algorithms based on deep learning that can acquire good performance on visual tracking problems. Usually, a convolutional neural network, for example, the VGG-M, is used to obtain the complex features representations of the video considering the robustness as well [1]. Compared with hand-craft methods, features learned by CNNs showed great generalization capability in the datasets. A multi-domain learning model (MDNet) based on CNN is proposed, which can separate the independent information of multiple targets from the targets. Thus, it has outstanding performance in visual tracking tasks [2]. Stochastic gradient descent method is used in this algorithm to update network parameters. However, the random sampling strategy will make it very slow. In 2017, a reinforcement learning algorithm was combined with the CNNs to solve the object tracking problem [3]. Action-decision network (ADNet) learns the strategy of selecting the optimal action from the current state of the target to track the object. This algorithm is three times faster than existing trackers based on deep networks as well as the detection and tracking strategies. Instead of learning a series of discrete actions, the ‘Actor-Critic’ framework searches continuous action in one step, which makes it faster [4].
In this paper, an improved ‘Actor-Critic’ model is proposed, combined with the VGG-M network to accomplish the visual tracking task. Compared with the existing frameworks, the action space is modified to improve scale estimation. It solves the problem that the scale of target cannot be captured accurately and improves the precision of the tracker. Then the impact of different optimization methods is discussed, including improved action space (IAS) and advantage function (AF), in the performance of the tracker. Finally, an optimal method is found to apply in the visual tracking task.

2. Related Work

2.1 ACT Method
ACT aims to locate the target with continuous actions of moving the bounding box in a sequence of frames. Usually, visual tracking problems can be achieved through a sampling and verification pipeline. A series of candidate states are randomly generated around the possible object locations as potential choices in each frame. And the model would find out an optimal state of the object from all the candidates. As a consequence, the tracking process is time-cost and computation-consuming, considering a large number of sample verification. Compared to other methods like MDNet and ADNet, ACT reaches a faster and equally good performance at that time. It uses Actor-Critic networks to give the action and evaluation of action quality, respectively based on deep reinforcement learning.

In ACT work, Boyu et al. [5] define the state \( s = [x, y, h, w] \), as the tracking bounding box, where \( (x, y) \) is the central position and \( h, w \) stand for the height and width of the box, respectively. Action space here is defined by \( a = [\Delta x, \Delta y, \Delta s] \), as the movement of the bounding box, and \( \Delta x, \Delta y, \Delta s \) denote the change of central position and the scale, respectively.

For offline training, the algorithm is a modified deep deterministic policy gradient (DDPG) to efficiently train the Actor-Critic model for the tracking task. The core idea is using training sample pairs stored in buffer under the reinforcement learning rule to slowly update the Actor and Critic models by Bellman equation.

In continuous action RL problems, the huge action space always causes less efficient training process and difficulty gaining positive rewards when following a random exploration strategy. To solve this, ACT method proposed a post model initialization to utilize the supervised information from the first frame of each video to better understand the environment. To achieve this, it first samples M bounding box \( b_{m | m=1} \) near the ground truth and the corresponding actions \( a_{m | m=1} \). Then \( s_m = \phi(b_{m | F}) \) is used to get the observation state \( s_m \). They fine-tune the actor model with the adaptive moment estimation method by minimizing the L2 loss function:

\[
\min_{\theta} \frac{1}{M} \sum_{m=1}^{M} [\mu(s_m | \theta^\mu) - a_m]^2
\]

Furthermore, ACT replaces the original noise sampling from DDPG with a probabilistic expert decision guidance to supersede the exploration mechanism in reinforcement learning as noise sampling, which is not suitable for tracking task due to the huge action space. With a certain probability \( \epsilon \), expert decision guidance will replace the action generated by the Actor model, and the probability \( \epsilon \) will also be reduced during the training process.

The ACT method was the first attempt to exploit the continuous action for visual tracking and performs competitive results against many state-of-the-art methods.

3. Methodology

3.1 Overview
In this paper, we attempt to locate the target with continuous action of moving the bounding box in each frame in a sequence of frames. We adapt the ACT model and make our own improvement in order to expect a more precise bounding box as well as real-time performance. The details of our framework are explained in the following subsections.
3.2 Argument Definition

The task can be considered as a continuous decision-making problem, so this work follows the Markov Decision Process (MDP).

Basic MDP includes components $s \in S$, $a \in A$, $s' = f(s, a)$, $r(s, a)$ defined as followed. In this work, the tracker is the agent to interact in the environment of successive state $s_1, s_2, s_3, s_4, ...$ with actions $a_1, a_2, a_3, a_4, ...$ and rewards $r_1, r_2, r_3, r_4, ...$. Given a state $s_n$, the agent generates $a_n$ to get to a new state $s_{n+1}$.

- **Action**: relative motion of the bounding box
  $$a = [x_1, x_2, y_1, y_2]$$

The four coordinate points represent the position of the new bounding box.

As shown in Figure 1, $[(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)]$ gives the bounding box.

- **State**: observation image patch within the bounding box.
  $$s = [x, y, h, w]$$
  $(x, y)$: centre position
  $h$: height of the bounding box
  $w$: width of the bounding box

The per-processing function $s = \phi(b, F)$ is defined 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.

- **State Transition**: applying the action $a$ to the original bounding box $b$, the new bounding box:
  $$s' = [x', y', h', w']$$

Correspondingly,

- **the centre position**: $(x', y') = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$
- **height of the bounding box**: $h' = y_2 - y_1$
- **width of the bounding box**: $w' = x_2 - x_1$

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

To be sure the bounding box is moving properly, we set a limit for action $a$: $x_1 < x_2$, $y_1 < y_2$.

- **Reward**: localization accuracy improvement when transferring state $s$ into state $s'$ with action $a$, so
it can be defined as the overlap ratio of the new bounding box and the ground truth G.

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

(2)

3.3 Networks

In this paper we use the ‘Actor-Critic’ Network to accomplish the tracking task. The structure of the network is shown in Figure 2. And ‘a’ means Actor; ‘c’ means Critic.

We use the pre-trained VGG-M model to initialize both Actor and Critic networks. Both networks have a similar structure with 3 convolutional layers. Except for Actor network, the fully-connect (FC) layer 4 and 5 are combined with the ReLU operation, giving 512 output nodes, and the last FC layer generates a four-dimensional output. For Critic network, fully-connect layer 5 is used for training, which gives the four-dimensional action vector to obtain a Q-value for action evaluation according to the current state. Fully-connect layer 6 is used for testing, which gives a score.

![Figure 2. The overall framework of ACT. ACT method consists of an actor and critic networks.](image)

3.4 Offline Training

The actor-critic network is trained based on DDPG algorithm using a period of frames to update the critic and actor models. To stabilize learning, two target networks are created and will be slowly updated based on main networks.

For critic model, a Q-Value network takes the state and action as input and outputs Q-Value. Learn \( Q(s, a) \) using the Bellman equation. With the two target networks, we need to minimize the loss function for training, which is defined as follows,

\[ L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \]  

(3)

\[ y_i = r_i + \gamma Q'(s'_{i}, \mu'(s'|\theta^\mu'))(\theta^{Q'}) \]

For actor model, a policy network takes the state as input and outputs the exact continuous action. Update actor model using the sampled policy gradient by applying the chain rule, we need to maximize it to get the max Q-Values.

\[ \nabla_{\theta} J \approx \frac{1}{N} \sum_i \nabla_{\theta} Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_j)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i} \]  

(4)

The training process is shown as follows,

First, we use frame sequences \( f \) and ground truth \( g \) from the dataset as input. The expected output would be trained weights for Actor network. After initializing Critic \( Q(s, a) \), target network \( Q' \), Actor \( \mu \) and target network \( \mu' \), and replay buffer \( R \), we get to the training iteration and randomly select a batch of frame sequence \([f_k, f_{k+1}, f_{k+2}, f_{k+3} \ldots f_{k+T}]\) along with their ground truth
where $k$ is the start frame number and $T$ is the number of frames we choose.

**Algorithm 1: The proposed algorithm**

| for each $t = 2, T + 1$
| ---
| Obtain state $s_t$ according to state $s_{t-1}$ and $f_{k-1}$
| Select action $a_t$ according to current policy & exploration probability $\epsilon$;
| Execute $a_t$ as calculation in state transition and observe reward $r_t$ and the next state $s'_t$.
| Store transition $(s_t, a_t, r_t, s'_t)$ in R

After the for loop, we sample a random mini-batch of $N$ transitions $(s_i, a_i, r_i, s'_i)$ from R, then update critic $Q(s, a)$ by minimizing the loss function by Eq1 and update actor $\mu(s, \theta)$ using the sampled policy gradient by Eq2. Last update the target networks:

$$
\theta^Q \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q*} \\
\theta^\mu \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu*}
$$

The training will finish when the reward becomes stable.

### 3.5 Training Improvement

In reinforcement learning, the value function $V$ evaluates the quality of current state $S$. The action-state value function $Q$ evaluates the quality of taking action $A$ under state $S$. We define the advantage function $A$ as:

$$
A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)
$$

The advantage function $A$ evaluates the quality of taking actions in state $S$ relative to the average reward, that is, the advantage of taking action. The advantage function is to normalize $Q$ value on the “Value baseline.” It is conducive to reduce variance because too large variance is an important factor leading to overfitting.

Instead of using $Q$-function, we use the advantage function to train the networks to improve learning efficiency and make learning more stable.

### 3.6 Tracking Process

In the online tracking stage, the actor network is in charge of tracking, and the critic network is for classification, aiming to give a score based on the action the actor network generates. To make the tracker more suitable and efficient for current dataset, we fine-tune both models with ground truth in the first frame.

For actor model, we initialize it with the same sample method used in training improvement, and fine-tune the model using Adam method to minimize the L2 loss function Eq1.

For critic model $v(s|\theta^v)$, we create label $l_m$ for the m-th bounding box following the rule similar to the reward

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

With the label and image $\{s_m, l_m\}$, Adam method is used to minimize the loss function:

$$
\arg \min_{\theta^v} - \sum_{s \in S} p_+(s|v; \theta^v) - \sum_{s \in S_-} p_-(s|v; \theta^v) 
$$

(5)

where $S$ includes positive and negative samples, and $p$ is the foreground and background probabilities for a given state $s$.

During the tracking in t-th frame, we use the pre-processing function $s_t = \varphi(b_{t-1}, F_t)$, and then give the state to actor model resulting in the action $a_t$, from which we can obtain a new bounding box $b'_t$ and the current state $s'_t$ in frame $F_t$. The critic model can output a score based on the current state. Then we follow the rule:

$$
\text{Optimal bounding box} = \begin{cases} 
b'_t, \text{score} > 0 \\
\text{Argmax } m(b^m_t)_{l_m=0}, \text{score} < 0
\end{cases}
$$
Due to the appearance changes, we need to update the critic network occasionally. When the verification score is less than 0, we use negative and positive samples collected from 10 previous frames to update the model using Eq. (5).

4. Results

4.1 OTB Evaluation

After the training process, we evaluate the method on OTB-2015 benchmarks with selected sequences in OTB-100. We compare the real-time performance with this method and the baseline ACT tracker, which performs better than other competing trackers, including PTAV [4], CFNet [6], ACFN [7], SiameFC [8], ECO- HC [9], LCT [10], LMCF [11], Staple [12], DSST [13] and KCF [14]. In this part we evaluate our method and ACT based on both precision and success plots using One Pass Evaluation (OPE).

The precision plot is the percentage of frames that the distance between the center point of the target position estimated by the tracking algorithm and the center point of the ground-truth target under a given threshold. The standard is with the threshold of 20 pixels.

The success plot is the percentage of frames that the overlap ratio between the target position bounding box estimated by the tracking algorithm and the bounding of the ground-truth under a given threshold. The standard is with the threshold of 0.5.

• OTB-2015

OTB-2015 is the extended datasets of one of the most popular visual tracking benchmark OTB-2013, containing 100 video sequences known as OTB-100. 11 different challenging scenarios, such as Illumination Variation, Scale Variation, Occlusion, Deformation, Motion Blur, Fast Motion, In-Plane Rotation, Out-of-Plane Rotation, Out-of-View, Background Clutters, Low Resolution are included.

We compare the performance of our method, which is ACT+IAS+AF and without AF, to find out the influence of AF and ACT alone. Figure 3 and Figure 4 show that both methods with and without AF give better performance than the baseline. This improvement is due to the more precise scale change we adopt for the action space. The curves of ACT+IAS+AF and ACT+IAS almost coincide in both plots since AF accelerate the training process but does not improve the performance.

Figure 3. OTB-2015 Precision Plot
Figure 4. OTB-2015 Success Plot

As shown in Table 1, in dealing with 11 challenging attributes, our method improves the accuracy of ACT in every attribute in both precision and success plots.

| Attribute               | PR    | ACT | ACT+IAS | ACT+IAS+AF | ACT | ACT+IAS | ACT+IAS+AF |
|-------------------------|-------|-----|---------|------------|-----|---------|------------|
| Low Resolution          | 0.972 | 0.989 | 0.989   | 0.671      | 0.698 | 0.712   |
| Background Clutter      | 0.828 | 0.938 | 0.938   | 0.592      | 0.685 | 0.687   |
| Out of View             | 0.759 | 0.936 | 0.942   | 0.595      | 0.723 | 0.725   |
| In-Plane Rotation       | 0.838 | 0.891 | 0.9     | 0.568      | 0.621 | 0.628   |
| Fast Motion             | 0.802 | 0.925 | 0.923   | 0.622      | 0.706 | 0.71    |
| Motion Blur             | 0.747 | 0.881 | 0.894   | 0.591      | 0.69  | 0.698   |
| Deformation             | 0.803 | 0.878 | 0.89    | 0.502      | 0.579 | 0.587   |
| Occlusion               | 0.751 | 0.874 | 0.89    | 0.528      | 0.614 | 0.624   |
| Scale Variation         | 0.793 | 0.887 | 0.897   | 0.536      | 0.616 | 0.625   |
| Out-of-Plane Rotation   | 0.784 | 0.891 | 0.901   | 0.533      | 0.615 | 0.624   |
| Illumination Variation  | 0.799 | 0.932 | 0.929   | 0.516      | 0.607 | 0.612   |
| ALL                     | 0.826 | 0.917 | 0.924   | 0.57       | 0.65  | 0.657   |

4.2 Visualization Results
In this part, we choose three tracking examples to visualize how our method performs better than ACT.
As shown in the three examples above, the green bounding box is ACT; blue is ACT + IAS; the yellow one is ACT + IAS + AF, and the red denotes the ground truth bounding box.

In some simple object and background environment scenarios, as shown in Figure 7, every method is able to locate the tracking object in almost every frame. However, when the man in the video turns his face, ACT can only locate part of the face part but not the whole head, because it can only change the scale. Our method, compared to ACT, covers more target areas.

The situation in Figure 6 is also simple scenario but with distance change of the target, and all the methods can still handle. ACT has the same issue, which is locating part of the object. Our method ACT + IAS + AF keeps a good performance, and well adjust the length-width ratio change of the target because of our flexible action space.
Figure 8. Convergence speed with/without advantage function

In the complicated scenario with background and illumination changing, our method perfectly handles the tracking task. Both ACT + IAS and ACT + IAS + AF can follow and enclose the object in the bounding box. ACT somehow, taking the shadow in the glass as the object in every frame near the glass windows, fails to the tracking task.

To sum up, our method performs equally and better in most scenarios in the tracking task, and it is adaptive to the length-width ratio and scale change of the target. This improvement is due to the more precise action space we proposed.

To compare with our own, converge speed with AF is a little bit faster than the ACT method at the beginning. However, the result is very close actually.

5. Conclusion
In this work, we propose an improved Actor-Critic tracker for robust tracking. First, the proposed improved action space, which regresses the width and height individually, can obtain a tight bounding box with a more precise scale. Then, we adopt the advantage function to improve our training process and achieve a fast convergence speed. We test our method on OTB100 dataset, which validates the effectiveness of the proposed method.

Reference
[1] Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV. (2015)
[2] Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. (2016)
[3] Yun, S., Choi, J., Yoo, Y., Yun, K., Choi, J.Y.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR. (2017)
[4] Fan, H., Ling, H.: Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking. In: ICCV. (2017)
[5] Chen B, Wang D, Li P, et al. Real-time'Actor-Critic'Tracking[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 318-334.
[6] 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)
[7] Choi J, Jin Chang H, Yun S, et al. Attentional correlation filter network for adaptive visual
tracking[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4807-4816.

[8] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional siamese networks for object tracking. In: ECCV. (2016)

[9] Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ECO: efficient convolution operators for tracking. In: CVPR. (2017)

[10] Ma, C., Yang, X., Zhang, C., Yang, M.: Long-term correlation tracking. In: CVPR. (2015)

[11] Wang, M., Liu, Y., Huang, Z.: Large margin object tracking with circulant feature maps. In: CVPR. (2017)

[12] Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.H.S.: Staple: Complementary learners for real-time tracking. In: CVPR. (2016)

[13] Danelljan, M., Haïger, G., Khan, F.S., Felsberg, M.: Accurate scale estimation for robust visual tracking. In: BMVC. (2014)

[14] Henriques, J.F., Rui, C., Martins, P., Batista, J.: High. speed tracking with kernelized corre-lation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 37(3) (2015) 583–596