Action-Improved Actor-Critic Tracking for Accurate Object Tracking

Yangrun Hu1a, †, Ruikang Wu2b, † and Yanjun Gu3c, †

1School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China;  
2School of Arts and Sciences, Rutgers University, New Brunswick, NJ, 08854, U.S.;  
3School of Computer Science, Wuhan University, Wuhan, Hubei, 430079, China.  
ahuyangrun@sjtu.edu.cn  
bro551@scarletmail.rutgers.edu  
c2017301200328@whu.edu.cn  
†These authors contributed equally.

Abstract. Visual object tracking is a fundamental problem, which tracks the target in the whole sequence with the given object in the first frame. Numerous tracking frameworks have been proposed. Actor-Critic Tracking (ACT) is based on RL, which is a popular framework with real-time speed. However, ACT cannot handle scale estimation effectively, since it cannot change the ratio of target size. In this paper, we aim to improve the scale estimation of ACT and propose the Action-improved ACT(AIACT) model. This work is based on the basic framework of ACT. ACT builds the ‘Actor’ deep neural network to track the target and the ‘Critic’ network as an inspection module to improve the model. We add a new element to the bounding box of ‘Actor’ model to change the aspect ratio. The experiment result shows that our model achieves better accuracy than the previous model. Our method achieves 0.646 and 0.912 score of success rate and precision rate on OTB dataset, which means AIACT model can handle the scale variation of the target. Therefore, it will benefit more applications.

1. Introduction
Visual tracking is one of the most important components in video processing and computer vision nowadays, which has a large number of realistic applications, such as medical imaging, surveillance and behavior analysis. Although much progress has been made in the past decades, it remains a very critical issue due to many difficulties like illumination variation, background clutter, rotation, and so on.

In recent years, deep learning has been used in many fields, and breakthroughs have been achieved, especially in computer vision. Many deep-learning-based tracking algorithms (e.g., C-COT [1], SINT [2], STCT [3], TCNN [4], MDNet [5], ADNet [6], ACT [7]) have significantly improved the tracking performance. MDNet achieves significant performance in benchmarks such as VOT2015 by embedding the offline-trained VGG-M network into the partial filter framework, which randomly generates candidate samples and verifies these samples by using a CNN-based observation model. ADNet is a reinforcement-learning-based tracker, which improves the performance of MDNet by changing the search strategy to the iterative strategy. Since ADNet requires many iterative steps in every frame, it still cannot meet the real-time requirement. ACT uses a learning-based strategy with continuous actions...
based on the ‘Actor-Critic’ framework to predict only one continuous action using the ‘Actor’ model to tracked object in each frame, which achieves a better real-time performance.

However, ACT defaults that the length and width of the tracked object change proportionally, which is true in most cases. But in some cases, the shape of the tracked object changes significantly. This will make the performance worse. Based on this observation, we aim to solve the scale estimation problem in ACT.

We follow the framework of ACT, and improve its ‘Actor’ model by adding a new variable to change the length and width separately. We use ILSVRC VID [8] dataset for offline training and use OTB dataset [9] to test our model. The result of this experiment shows that the method is better than ACT when tracking targets whose aspect ratio changes significantly.

- We proposed an Action-improved ACT(AIACT) to handle scale estimation effectively by improving its action space.
- We test our method on OTB dataset and obtain 0.646 and 0.912 scores of success rate and precision rate, which outperforms the baseline method.

2. Method

2.1. Overview of ACT
There are two key components of the Actor-Critic network: Actor and Critic. For Actor network, the optimization method is policy gradients, so Actor network can easily provide the continuous offset between two frames. For the Critic network, it depends on deep reinforcement learning and the predecessor is Q-learning, so it can update the action for each step. Compared to Q-learning, policy gradients update the action for each round, which is less efficient than Q-learning. Besides, the Critic network aims to distinguish whether the candidate is the target or not. In general, the Actor-Critic network gets the benefit from policy gradient and Q-learning.

![Figure 1. The pipeline of the proposed tracking algorithm.](image)

2.1.1. Model/framework. **Actor-Network** For this network, the main idea is to get the change of the target compared to the previous frame. Network Input: The picture or target needs to be tracked. Network Output: The offset of the current frame relative to the previous frame. We use $\Delta x, \Delta y, \Delta s$ to depict the relative motion of the tracked object.

**Critic Network** Depending on the output given by the actor-network, the critic network will determine whether the offset is correct or not. Network Input: the target for the current frame. Network Output: we get a verification score.

**Network Structure** In the pre-trained process, we use the VGG-M model to initialize 'Actor' and 'Critic' networks. Both two networks contain three convolutional layers. For the 'Actor' network, the following two fully connected layers have 512 output nodes connected with the ReLU operation. Besides, a three-dimensional output is produced by the last fully connected layer. For offline training, the "Critical" model has a similar structure to the "Actor" model, except that the last layer is fully
connected because it needs to connect the 3D action vector and obtain a Q value based on the current state to evaluate the action.

There are three essential elements in the reinforcement learning: action, state, and reward. In the following article, we discuss these three elements.

Our method is based on Markov Decision Process (MDP), which is used for solving the sequential decision problem when tracking. There are four components in MDP. S is a finite state set that contains all possible states of the agent and the environment. A is an action set that contains all possible actions of an agent. P is the state transaction function. \( P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a) \), which means the probability of the action made by the agent at time t causes the state of the Markov process to shift from s at time t to s' at time t+1. \( R_a(s, s') \) is the real-time reward that an agent receives by moving the state from S to S’ through action A. Here, we detail the four components in our model.

Action, state, transaction and reward are also the essential elements in reinforcement learning, so we will discuss how we implement four elements specifically in the following paper.

**State** The state S is defined as the observation image patch in the surrounding box \( b=[x, y, h, w] \). \((x,y)\) represents the center position, \( h \) represents the height and the \( w \) represents the width. Besides, the pre-processing function is also defined as \( s = \phi(b, F) \). This function is used for trimming the image patch in the bounding box \( b \) within a given frame \( F \) and resizes it to fit the input size of the deep network.

**Action and transaction function** As we mentioned in the actor-network, \( \Delta x, \Delta y, \Delta s \) have been adopted to depict the relative motion of the tracked object. For \( \Delta x, \Delta y \), they represent the relative horizontal and vertical translations. For \( \Delta s \), it represents for the relative scale change. We made some constraints to limit these elements. \(-1 \leq \Delta x \leq 1, -1 \leq \Delta y \leq 1, \text{and} -0.05 \leq \Delta s \leq 0.05\). We define \( a=[\Delta x, \Delta y, \Delta s] \) and apply the action \( a \) to a bounding box \( b \). Then we can get a new bounding box \( b' \). As a result, \( b'=[x', y', h', w'] \).

\[
\begin{align*}
x' &= x + \Delta x \times h \\
y' &= y + \Delta y \times w \\
h' &= h + \Delta s \times h \\
w' &= w + \Delta s \times w 
\end{align*}
\]

Besides, we can use the pre-processing function \( s' = \phi(b', F) \) to get the state transition process \( s' = f(s, a) \).

**Reward**: \( r(s, a) \) is the reward function that describes the improvement in positioning accuracy when the state \( s \) is transformed into state \( s' \) for a given action \( a \). We define the reward function as following way.

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

2.1.2. Offline training. We use the DDDP approach to train our 'Actor-Critic' network. The main idea is to iteratively update the model of "critics" and "actors" to collect training sample pairs based on the RL rule. For the 'critic' model, Q(s,a) can be learned by using the Bellman equation for given n pairs of \((s_i, a_i, r_i, s'_i)\). Besides, based on the model parameters, the chain rule is applied to the expected return of the initial distribution to update the "actor" model.

**Training Process Improvement** Because the action space was very large in our tracking problem, we cannot use the DDDP approach directly. As a result, we made two improvements to the training process.

- Because the action space is too huge to obtain a positive reward, during the training model, it makes the DDPG approach less effective. In order to solve this problem, the supervised information from the first frame is used to initialize the "actor" model to fit the current environment. In other words, the "Actor" model is fine-tuned by an adaptive moment estimation method to minimize the following L2 loss function, Where the number of training samples is M. ‘Actor’ network is represented by \( \mu \). \( s_m \) is the m-th sampled state and \( a_m \) is ground truth action.
\[
\min \frac{1}{M} \sum_{m=1}^{M} [\mu(s_m|\theta^t) - a_m]^2
\]

- Based on what we got from the scheme (1), it cannot completely solve the imbalance between positive and negative samples. The reason is that there are many unpredictable challenges leading to the tracking drift. Therefore, we use expert decisions to guide the learning process. We used probabilistic expert decision guidance to replace the exploration mechanism in reinforcement learning. In a video sequence, \( q \) at a certain probability, the expert decision-making guidance instead of the "actors" network output action. In the process of training, probability \( q \) gradually reduced.

2.1.3. Online tracking. **Network Initialization.** First, we initialize the "actor" and "critic" models with the ground truth in the first frame. For the 'actor' network, we start at sampling \( M \) candidate bounding boxes and calculating corresponding accurate actions. We use the pre-processing function to extract the image observation \( s_m \) for the candidate location \( b_m \).

For the 'critic' model, we use the following rule to assign a binary label \( l_m \) to the \( m \)-th candidate,

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

Where \( G \) is the ground truth bounding box. Besides, we also use the Adam method to initialize the 'critic' network. The output of the 'critic' network is foreground and background probabilities for a given state.

**Tracking via 'Actor-Critic'.** We utilize both the "actor" and "critic" networks in our tracking-and-validation scheme for online tracking. First, we start by using the pre-processing function to calculate the state \( s_t \). Second, we get the action by putting the state \( s_t \) to the 'actor' network. In other words, the information of the current frame or the target position of the previous frame is the input for the actor-network. The result is getting the action. Then, we use the 'Critic' network to check the observation based on score. If the score is greater than 0, which represents the action was trustable and used the location \( b'_t \) as the optimal location in the t-th frame. If the score is less than 0, we employ a redetection technique that uses a "critical" network to get a series of sampled candidates. The candidate who gets the highest score will produce the optimal location \( b'_t \). The improved new bounding box becomes

\[
b' = [x', y', h', w']
\]

\[
\begin{aligned}
x' &= x + \Delta x \cdot h \\
y' &= y + \Delta y \cdot w \\
h' &= h + \Delta h \cdot h \\
w' &= w + \Delta w \cdot w
\end{aligned}
\]
2.3. Implementation Details
The dataset used is ILSVRC VID [8]. The information of our machine is CPU: 3.6 GHz and i9-9900k, GPU: Nvidia RTX 2080ti.

**Offline Training.** For training, the epoch is 250000. We use 768 video sequences from the dataset and randomly pick continuous twenty to forty frames in a video for each iteration. The learning rate of the ‘Actor’ and ‘Critic’ network is 1e-6 and 1e-5.

**Online Tracking.** In the first frame, 500 positive and 5000 negative samples are collected with ground truth. We train the ‘Actor’ network using only positive samples. For initialization, the learning rates of the ‘Actor’ and ‘Critic’ network are both 1e-4. The batch size of the ‘Actor’ network is 64, and the batch size of the ‘Actor’ network is 128. If the highest score of all candidates is smaller than 0, we treat it as a tracking failure.

3. Results and Discussion

3.1. Dataset Description
We test our model on the OTB dataset [9], which contains 100 videos. And we select 40 videos for testing and validate the effectiveness of scale variation handling.

Since many factors that can make the tracking process very challenging, such as background clutters and in-plane rotation, the OTB dataset classifies the sequences into 10 attributes which affect the tracking result significantly in some cases. This can help us analyze the algorithm’s advantages and weaknesses. The definitions of these 10 attributes are as follows.
- Illumination Variation (IV): The illumination changes significantly in the target region.
- Occlusion (OCC): The target is occluded partly or completely.
- Deformation (DEF): Non-rigid object change obviously deforms.
- Motion Blur (MB): The motion of the object or camera makes the target motion.
- Fast Motion (FM): The ground truth moves faster than normal speed.
- In-Plane Rotation (IPR): The object rotates in the image plane.
- Out-of-Plane Rotation (OPR): The object rotates out of the image plane.
- Out-of-View (OV): Part of the targets move out of the view.
- Background Clutters (BC): The background’s color or texture is similar to the object.
- Scale Variation (SV): The ratio of the bounding boxes changes extremely

This model aims to promote SV tracking performance. But the result demonstrates that AIACT performs better than ACT in all these ten attributes.

We use precision and success plots to evaluate the model. The precision plot shows the percentage of the frames which has a smaller deviation between the ground truth and the tracking location than a given threshold. The success plot illustrates the Intersection-Over-Union of the tracking box and the ground truth is higher than a given ratio.

3.2. Experimental Result
Figure 2 reports the precision and success plots over 40 videos in OTB-2015. The figure shows that our AIACT method achieves better performance than ACT in terms of precision and in terms of success. These results are attributed to the contribution of the added element.

Table 1 indicates that the results on sequences belong to scale variation attributes between AIACT and ACT method. In the tasks of object tracking with scale variation, our method achieves 0.880 precision and 0.612 success, which are both better than ACT does. The results demonstrate that our proposed method has a considerable improvement on scale variation compared with ACT.
Figure 2. The precision plot and success plot of the proposed method and ACT.

Table 1. The attribute-based comparison on scale variation between AIACT and ACT.

| Methods | Precision Rate | Success Rate |
|---------|----------------|--------------|
| AIACT   | 0.880          | 0.612        |
| ACT     | 0.784          | 0.529        |

3.3. Visualization Result
Figure 3 describes the visualization result of our baseline and proposed method. Our algorithm achieves more accurate tracking results than ACT in Dudek sequence and FleetFace sequence which contain many cases of scale variation. From the figure, we can see that the bounding boxes draw by AIACT (the green boxes) are much closer to ground truth (the red boxes). The results demonstrate AIACT’s robustness to scale variation.

4. Conclusion
In this paper, we propose an AIACT method to improve the ability of scale variation handling for ACT method. Based on the original ‘Actor’ and ‘Critic’ networks, we use two variables which separately control length and width to replace the original variable that only controls the scale. We train the model offline on ILSVRC VID dataset. Our AIACT achieves 0.646 of success rate and 0.912 of precision rate on OTB dataset, which is better than the original ACT model. The advantages of this method can be summarized as follows. Firstly, this method can solve the scale estimation problem of ACT effectively by the proposed improved action space. Secondly, this method is more suitable for actual tracking scenes, which contains target’s scale variation and has more applications.
**Figure 3.** The visualization results of AIACT and ACT trackers.

**References**

[1] M. Danelljan, A. Robinson, F.S. Khan, M. Felsberg, Beyond correlation filters: learning continuous convolution operators for visual tracking, in: ECCV. (2016)

[2] R. Tao, E. Gavves, A.W.M. Smeulders, Siamese instance search for tracking, in: CVPR. (2016)

[3] L. Wang, W. Ouyang, X. Wang, H. Lu, STCT: sequentially training convolutional networks for visual tracking, in: CVPR. (2016)

[4] H. Nam, M. Baek, B. Han, Modeling and propagating cnns in a tree structure for visual tracking, Clin. Orthopaedics Related Res. (2016). arXiv:abs/1608.07242.

[5] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: CVPR. (2016)

[6] S. Yun, J. Choi, Y. Yoo, K. Yun, J.Y. Choi, Action-decision networks for visual tracking with deep reinforcement learning, in: CVPR. (2017)

[7] B. Chen, D. Wang, P. Li, S. Wang, H. Lu, Real-time ‘Actor-Critic’ Tracking, in: ECCV. (2018)
[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M.S. Bernstein, A.C. Berg, F. Li, Imagenet large scale visual recognition challenge, in IJCV. (2015)

[9] Y. Wu, J. Lim, M.H. Yang, Object Tracking Benchmark, in: PAMI. (2015)