Deep Reinforcement Learning with Iterative Shift for Visual Tracking

Liangliang Ren[1,*], Xin Yuan[1,*], Jiwen Lu[1,†], Ming Yang[2], Jie Zhou[1]

[1]Tsinghua University; [2]Horizon Robotics, Inc.
{renll16, yuanx16}@mails.tsinghua.edu.cn; {lujiwen, jzhou}@tsinghua.edu.cn; ming.yang@horizon-robotics.com

Abstract. Visual tracking is confronted by the dilemma to locate a target *both* accurately and efficiently, and make decisions *online* whether and how to adapt the appearance model or even restart tracking. In this paper, we propose a deep reinforcement learning with iterative shift (DRL-IS) method for single object tracking, where an actor-critic network is introduced to predict the iterative shifts of object bounding boxes, and evaluate the shifts to take actions on whether to update object models or re-initialize tracking. Since locating an object is achieved by an iterative shift process, rather than online classification on many sampled locations, the proposed method is robust to cope with large deformations and abrupt motion, and computationally efficient since finding a target takes up to 10 shifts. In offline training, the critic network guides to learn how to make decisions jointly on motion estimation and tracking status in an end-to-end manner. Experimental results on the OTB benchmarks with large deformation improve the tracking precision by 1.7% and runs about 5 times faster than the competing state-of-the-art methods.

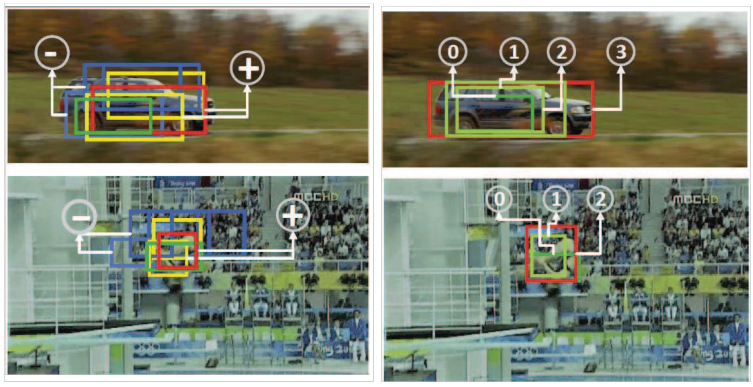
Keywords: Visual object tracking, reinforcement learning, actor-critic algorithm

1 Introduction

Visual object tracking (VOT) aims at locating a target efficiently in a video sequence, which remains a challenging problem in unconstrained applications due to deformation, abrupt motion, occlusions and illumination, after several decades of intensive research [5, 10, 20, 36, 41, 42, 51]. Essentially VOT needs to address 3 key issues: 1) How to represent a target, *i.e.*, the observation model; 2) How to *efficiently* leverage the motion smoothness assumption to locate a target in the next frame; 3) How to update tracking models online, if necessary, to handle dynamic scenarios.

The appearance models have evolved from intensity templates [19], color histograms [14], and sparse features [4], to the dominating *deep features* [47] extracted by CNN models. Thus, naturally tracking may be formulated as a

* Indicates equal contribution.
† Corresponding author.



(a) Classification based methods (b) Iterative shift based method

Fig. 1: Illustration of tracking using classification (left column) vs. iterative shift (right column): tracking a fast moving vehicle (first row) and tracking a diving athlete with large deformation (second row). Given the initial box (green), classification based methods sample many proposals, select the box (red) with the highest classification score, and collect positive (yellow and red) and negative samples (blue) to fine-tune the classifiers online. There may not be enough good samples for online learning in these hard scenarios. In contrast, the proposed iterative shift tracking adjusts the bounding box step by step to locate the target (*e.g.*, 3 steps for the vehicle and 2 steps for the athlete), and makes decisions formally when and how to update object models by reinforcement learning. The shift process generally tends to be more efficient since less candidate regions are evaluated than in classification based methods

classification or detection-and-association problem [35] using CNN classifiers. Even a strong observation model may not capture all possible variations of targets and need to be updated on-the-fly during tracking. Nevertheless, online classifier learning may be vulnerable to samples with ambiguous labels in hard scenarios, such as deformation, quick motion and occlusions, *etc.*, leading to *model drift*. The tracker needs to make decisions simultaneously on target's motion status and on tracking status, *i.e.*, whether and how to update observation models or even restart tracking if necessary. These are indeed tough decisions to make during online tracing.

To tackle the aforementioned issue 2 and 3, we introduce a deep reinforcement learning process to make decisions jointly on a target's motion status and a tracker's status in VOT. The motion status, *i.e.*, the displacement and scaling of an object's bounding box, is estimated in an efficient iterative shift process by a *prediction* network. The tracker's status, referring to whether or how to update the observation model and whether stop and restart tracking, is determined by an *actor* network. The proposed method, coined as deep reinforcement learning with iterative shift (DRL-IS), exploits the correlation between object's motion

estimation and current tracking status. The prediction and actor networks are learned offline from a large number of training video sequences guided by a *critic* network, on how to take actions given the current frame and the previous target location and representations.

This method utilizes reinforcement learning as a principled way to learn how to make decisions during tracking, therefore, it is especially robust to deal with hard cases such as deformation or abrupt motion, where either updating the model or stop-and-restart may be a sensible action. In contrast, existing methods ADNet [52], EAST [21] and POMDP [44] which employed reinforcement learning to either estimate motion or make decisions on tracking status separately. Moreover, as shown in Fig. 1, the tracking result is estimated iteratively, instead of performing CNN classification on many candidate locations, thus leading to an efficient computation.

The main contributions of our paper are on two-fold: 1) We propose an Actor-Critic network to predict the object motion parameters and select actions on the tracking status, where the rewards for different actions are dedicatedly designed according to their impacts; 2) We formulate object tracking as an iterative shift problem, rather than CNN classification on possible bounding boxes, thus locates a target efficiently and precisely. The proposed DRL-IS is particularly capable of dealing with objects with large deformations and abrupt motion, since the motion parameters are iteratively estimated and accumulated by the prediction network, and in such hard cases the tracker is kind of self-aware to update the target feature and model or resort to detection to restart tracking. Our tracker achieves 0.909 distance precision, 0.671 overlap success on the OTB2015 benchmark and 0.812 distance precision and 0.590 overlap success on the TempleColor128 benchmark, on a par with the best performance, and runs about 5 times faster than competing state-of-the-art methods.

2 Related Work

Visual tracking has undergone extensive study over several decades on how to represent and locate a target in video sequences, and adapt the observation model online if necessary. Deep neural networks, pre-trained for recognition tasks, tend to be also effective in delineating an object appearance in tracking, *e.g.*, as in the MDNet [35], FCNT [46], and CREST [42] trackers, and [10, 18, 32, 47]. To find a target in current frame, a motion model is assumed to sample some candidate locations, as in the Kalman filter [1] or particle filter [22, 38]. Then, the observation model may be evaluated on hundreds of these locations, as a correlation filtering in MOSSE [5] and KCF [20], or as a discriminative classification [11] or regression problem [16], which is demanding in computation. Alternatively, an observation model may allow to calculate or search the candidate locations gradually and iteratively, as in the optical flow [14] or mean-shift tracking [9], which is generally efficient since only a few locations examined. This motivates us to propose the iterative shift process, where a prediction network

adjusts target locations in an iterative manner and evaluates the neural net much less times.

The observation model may need to be updated during tracking to follow the changing appearance of an target, for instance, by collecting positive and negative samples [24] or bags [3] to conduct online learning [50]. A tracker has to make very tough decisions on when and how to update the observation model. For some difficult scenarios, such as deformation, occlusion and abrupt motion, on one hand, without any model update, the tracker may lost the target, on the other hand, due to some ambiguous or wrong labels, the tracker may drift to clutter background after the online update. In these hard but not rare cases, a sensible decision might be to stop tracking and resort to object detection or other means to reinitialize, rather than drifting blindly and silently. This fundamental issue demands for a formal decision making procedure in tracking.

Deep reinforcement learning [2,6,7,23,26,29,33,34,40] is a principled paradigm to learn how to make decisions and select actions online, which has achieved great successes in Atari games [34], search of attention patches [7], and finding objects [29] and visual relations [40]. Recently, reinforcement learning has been adopted for tracking [21, 25, 44, 52, 53], *e.g.*, an action-decision network [52] to generate actions to seek the locations and the sizes of a target object, or a decision policy tracker [44] by using reinforcement learning to decide where to look in the upcoming frames, and when to re-initialize and update its appearance model for the tracked object. In this paper, we extend to learn how to jointly derive the target motion and make decisions on the tracker status, by a new and unified actor-critic network.

3 Approach

The proposed deep reinforcement learning with iterative shift (DRL-IS) approach involves three sub-networks: 1) the actor network, 2) the prediction network, and 3) the critic network, which *share* the convolutional layers and one fully connected layer (fc4), as shown in Fig. 2. We elaborate the formulation of DRL-IS for tracking and the learning procedure of these networks, in the following subsections.

3.1 Iterative Shift for Visual Tracking

We formulate visual object tracking as an iterative shift problem. Given current frame and previous tracking results, the prediction network ψ iteratively shifts the candidate bounding box to locate the target, meanwhile, the actor network θ makes decisions on the tracking status, whether or not to update the target representation and the prediction network, or even restart tracking.

Formally, given a video $V = \{I_1, I_2, \cdots, I_N\}$, where I_t is the tth frame. The tracker is initialized by cropping a target with $l_1 = \{x_1, y_1, w_1, h_1\}$ in the first frame and its appearance is represented by the feature f_1 , *i.e.*, the fc4 layer's outputs in the shared network. With the tracking results of $l^*_{t-1} =$

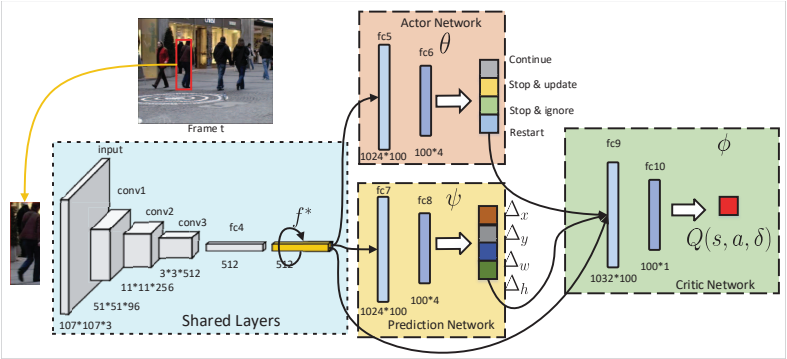


Fig. 2: The overview of the DRL-IS tracking method. Given the initial bounding box of a target, we first extract deep feature $f \in \mathbb{R}^{1*512}$ from fc4 layer. Then we concatenate the the feature of a candidate box f and the current target feature $f^* \in \mathbb{R}^{1*512}$. We generate shift δ using the prediction network ψ and employ the actor network θ. For action *continue*, we adjust the bounding box of the target according to the output δ of ψ. For action *stop* and *update*, we stop the iteration and update the appearance features of the target and the parameters of ψ, while we skip the update for action *stop* and *ignore*. When taking action *restart*, the target may be lost, so we re-sample for the initial bounding box. In the training stage, we use a deep critic network to estimate the Q-value of current actions with δ, and fine-tune the prediction network ψ and actor network θ

$\{x_{t-1}, y_{t-1}, w_{t-1}, h_{t-1}\}$ and f^*_{t-1}, we first extract f_t of I_t cropped by l^*_{t-1}, and exploit the prediction network ψ to predict the movement δ of the target between frames, which takes f_t and f^*_{t-1} as input:

$$\delta = \psi(f_t, f^*_{t-1}). \tag{1}$$

We denote the outputs of the prediction network as $\delta = \{\Delta_x, \Delta_y, \Delta_w, \Delta_h\}$:

$$\begin{aligned} \Delta_x &= (x_t - x_{t-1})/w_{t-1}, \\ \Delta_y &= (y_t - y_{t-1})/h_{t-1}, \\ \Delta_w &= log(w_t/w_{t-1}), \\ \Delta_h &= log(h_t/h_{t-1}), \end{aligned} \tag{2}$$

where Δ_x and Δ_y specify a scale-invariant translation of the bounding box, Δ_w and Δ_h specify log-space translations of the width and height of bounding box against the previous frame [17]. It is hard to estimate the movement and shape change of the target accurately in one step when the object moves rapidly or deforms. Hence, the prediction network outputs the adjustments of the bounding box iteratively and accumulate them to obtain the tracking result. Thus, the neural network is evaluated in K_t iterations at I_t and δ_k of each step in Eq. 2 are accumulated. This iterative shift process is considerably faster than running a classification network on hundreds of bounding boxes.

Meanwhile, the tracking status may affect the results as well, *e.g.*, updating the prediction network on the fly if necessary. To make decisions jointly on a target's motion status and a tracker's status, we use the actor network θ to generate the actions $a_1, a_2, \cdots, a_k, \cdots, a_{K_t}$ according to a multinomial distribution:

$$p(a|s_{t,k}) = \pi(s_{t,k}|\theta), \sum_i p(a_i|s_{t,k}) = 1, \tag{3}$$

where $a_k \in \mathcal{A} = \{continue, stop\ \&\ update, stop\ \&\ ignore, restart\}$, and the initial state $s_{t,0} = \{I_t, l_{t,0}, f^*_{t-1}\}$ contains the image I_t, initial location $l_{t,0} = l^*_{t-1}$, and the appearance feature f^*_{t-1}, and $\pi(s_{t,k}|\theta)$ derives from the outputs of the actor network θ.

For the action *continue* (continue shifting without updating the model) in step k, the shift $\delta_k = \psi(f_{t,k}, f^*_{t-1})$ is generated by the prediction networks ψ. $f_{t,k}$ is extracted from the crop $l_{t,k}$. The position, $l_{t,k}$, of the target is updated iteratively according to δ_k with $l_{t,k-1}$.

For the action *stop & update* (stop shifting and update the model), we stop the iterations and take $l^*_t = l_{t,K_t}$ as the location for object and update the feature of the target and the parameters of the prediction network ψ,

$$f^*_t = \rho f_{t,K_t} + (1-\rho) f^*_{t-1}, \tag{4}$$

$$\psi_t = \psi_{t-1} + \mu \mathbb{E}_{s,a} \frac{\partial Q(s,a,\delta|\phi)}{\partial \delta} \frac{\partial \delta}{\partial \psi}, \tag{5}$$

where ρ is a weight coefficient since Eq.(5) is a common practice in tracking allowing the target feature evolve as a weighted sum of current and previous representations. Eq.(6) is an online learning rule to update the prediction network, so μ is an adequate learning rate. $Q(s, a, \delta)$ is the output of critic network ϕ and defined in Eq. 11. This action indicates a reliable tracking, confident enough to update the target representation and the model.

For the action *stop & ignore* (stop shifting without updating the object feature), we stop the iteration and take $l^*_t = l_{t,k}$ as the location for object and move on to track the target in the next frame, where the appearance feature f^*_t and the prediction network ψ are not updated. This action indicates that the target is found, yet the tracker is not confident to update the model, *e.g.*, if motion blur or occlusions present.

For the action *restart* (restart tracking), we restart the iterationby re-sampling a random set of candidate patches L_t around l^*_{t-1} in I_t, and select the patch which has the highest Q-values, which is defined in Eq. 12 according to the IoU objective, as the initial location:

$$l_{t,0} = \arg \max_{s=\{I_t, l, f^*_{t-1}\}, l \in L_t} Q(s, a = stop\ \&\ update, \delta = 0|\phi). \tag{6}$$

This action represents the cases that the tracker loses the target temporarily and resorts to an extensive search to re-initialize tracking.

Fig.3 presents a sample action sequence in tracking. The prediction and actor networks formulate the motion estimation and tracking status change in a unified

way as taking actions in reinforcement learning. Nevertheless, learning these neural networks requires dedicatedly designed rewards for each type of actions.

3.2 Training the neural networks in DRL-IS

In this subsection, we detail the training procedure of the prediction, actor, and critic networks by deep reinforcement learning, from a large number of labeled video sequences. Note that the prediction network is pre-trained offline while during online tracking, both the prediction and actor networks are jointly updated by the actor-critic approach.

Learning of the prediction network: The prediction network estimates the iterative shift of the object in a given frame, from the object location and features in consecutive frames. We pre-train a convolutional neural network in an end-to-end manner to predict the shift of the target object between frames or iteration steps.

Network Architecture: As illustrated in Fig. 2, the prediction network uses three convolutional layers to extract features from the target patch and the current candidate box during pre-training. Then the features are concatenated and fed into two fully connected layers to produce the parameters which estimate the location translation and scaling changes.

Network Inputs: We sample pairs of crops from the sequences between every two frames to feed the network. The first crop is the object location in the previous frame and the second crop is in the current frame at the same location. The crops are padded with a fixed ratio to the object scale, which is empirically determined in our experiments. The network receives a pair of crops which are warped into 107×107 pixels and estimates the motion δ between two adjacent frames.

Network Pretraining: Instead of extracting the feature of the region proposals and performing regression on the bounding box, we train a fully end-to-end network to learn location translations and deformations directly. We perform data augmentation by sampling multiple examples with scale variations which are near the target bounding box and then create crops in the current frame. Using labeled video frames and these augmented samples, the training of prediction network promotes to locate a target with less iteration steps.

DRL-IS with Actor-Critic: We exploit the actor-critic algorithm [28] to jointly train the three sub-networks, θ, ψ, ϕ. Firstly, we define the rewards according to the tracking performance. The reward of the action *continue* with $\delta_{t,k}$ is defined by ΔIoU rather than the IoU to adjust bounding boxes.

$$r_{t,k} = \begin{cases} 1 & \Delta_{IoU} \geq \epsilon \\ 0 & -\epsilon < \Delta_{IoU} < \epsilon \\ -1 & \Delta_{IoU} \leq -\epsilon \end{cases}, \tag{7}$$

where $\epsilon > 0$ and Δ_{IoU} is computed as:

$$\Delta_{IoU} = g(l_t^*, l_{t,k}) - g(l_t^*, l_{t,k-1}), g(l_i, l_j) = \frac{l_i \cap l_j}{l_i \cup l_j}. \tag{8}$$

For the action *stop* & *update* and *stop* & *ignore*, the rewards are defined by the IoU of the final prediction and the ground truth. To encourage tracking stop with less iterations, the positive reward is related to the the iteration times K_t. We take l_t^* as the location for object and the rewards are computed as:

$$r_{t,K_t} = \begin{cases} 10/K_t & g(l_t^*, l_{t,K_t}) \geq 0.7 \\ 0 & 0.4 \leq g(l_t^*, l_{t,K_t}) \leq 0.7 \\ -5 & else \end{cases}. \tag{9}$$

For the action *restart*, the reward is positive when the IoU of the final prediction and the ground truth is less than 0.4 considering the high computational costs of *restart*.

$$r_{t,K_t} = \begin{cases} -1 & g(l_t^*, l_{t,K_t}) \geq 0.7 \\ 0 & 0.4 \leq g(l_t^*, l_{t,K_t}) \leq 0.7 \\ 1 & else \end{cases}. \tag{10}$$

Then we define the calculation of Q-values of each action. The Q value of the action *continue* and other actions are quite different, since the reward of *continue* is based on the increment of IoU while others are based on the tracking performance evaluated by IoU. The Q value of action *continue* with $\delta_{t,k}$ is computed as follows:

$$Q(s, a, \delta_{t,k}) = \sum_{i=k}^{K_t} \gamma^{(i-k)} r_{t,i}. \tag{11}$$

The Q values of actions *stop* & *update*, *stop* & *ignore*, *restart* are computed as:

$$Q(s, a, \delta_{t,k} = 0) = \sum_{j=t}^{N} \gamma^{j-t} r_{j,k_j}. \tag{12}$$

Eq.(12) sums the rewards upon the step k in the current frame while Eq.(13) sums the rewards upon the time step t. The reason for the different calculations of Q-values in Eq.(12) and Eq.(13) is that the action *continue* locates the target with the current models in frame t while other actions involve the decision whether to stop tracking based on previous tracking performance.

Finally, we formulate the optimization problem of ϕ and θ as follows:

$$\phi = \arg\min_{\phi} L(\phi) = \mathbb{E}_{s,a}(Q(s, a|\phi) - r - \gamma Q(s', a', |\phi^-))^2, \tag{13}$$

$$\theta = \arg\min_{\theta} J(\theta) = -\mathbb{E}_{s,a} \log(\pi(a, s|\theta))\hat{A}(s, a). \tag{14}$$

s' is the next state and $a' = \arg\max_a Q(s', a|\phi^-)$. Action-value $\hat{A}(s, a)$ and value function $V(s)$ is calculated as follows:

$$\hat{A}(s, a) = Q(s, a|\phi) - V(s), \tag{15}$$

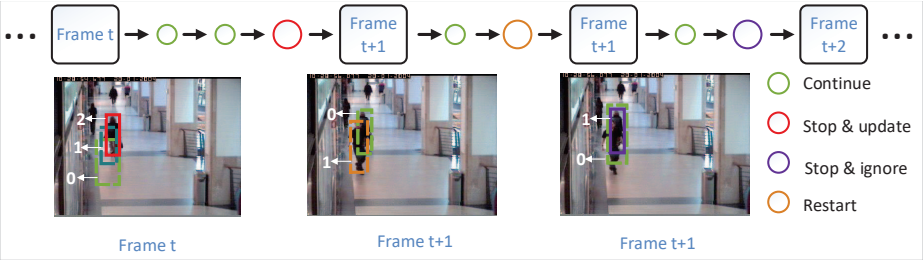


Fig. 3: An illustrative example of the actions on tracking status change by the actor network: 1) at I_t, the target is readily located by two *continue* actions and a *stop & update* action updates the target feature $f*_t$ and the prediction network ϕ accordingly; 2) at I_{t+1}, at first, a *continue* action tracks to a distractor person nearby, than the tracker spots this and take a *restart* action to re-initialize the tracking; 3) the shift process is restarted at I_{t+1}, with a *continue* action, the target is found yet the scale is not reliable, and then a *stop & ignore* action return the results but does not update the target feature f_t^*

$$V(s) = \mathbb{E}_s \pi(s, a|\theta^-) Q(s, a|\phi^-), \tag{16}$$

where ϕ^- is the target network, which has the same architecture with ϕ but is only updated in each 10 iterations. Please refer to [37] for the details of reinforcement learning. We update the parameters of the critic network ϕ and actor network θ as follows:

$$\phi = \phi - \mu_\phi \frac{\partial L(\phi)}{\partial \phi}, \tag{17}$$

$$\theta = \theta - \mu_\theta \frac{\partial J(\theta)}{\partial \theta}. \tag{18}$$

Algorithm 1 summarizes the learning of proposed method.

4 Experiments

To validate the proposed approach, we conducted experiments on the popular Object Tracking Benchmark [48, 49], Temple-Color128 [31] and VOT-2016 [30], and compared with recent state-of-the-art trackers.

4.1 Datasets and Settings

We conducted experiments on the standard benchmarks: OTB-2015, Temple-Color128 and VOT-2016. OTB-2015 [49] contains 100 video sequences, where each video was fully annotated with ground truth bounding boxes. Temple-Color128 contains 128 color sequences. The challenging attributes for visual object tracking on these two datasets include illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB),

Algorithm 1 : The training of networks in DRL-IS

Input: Training set: $\mathbf{V} = \{V_i\}$, ψ, and convergence error ϵ_1, maximal iterations M.
Output: ϕ, θ and ψ

Initialize ϕ and θ;
for all $m = 1, 2, \ldots, M$ **do**
  Randomly select a video V;
  Initialize the appearance feature f and l_1 using the ground truth in 1-st frame
  for all $t = 2, 3, \ldots, N$ **do**
    Generate the action a using θ;
    while $a ==$ *continue* **do**
      compute δ using ψ;
      Adjust $l_t = l_t + \delta$
      Generate an action a using θ;
    end while
    Update ψ, f_t^* or restart according to a;
  end for
  Calculate $J_t(\theta)$ and L_ϕ;
  Update actor network θ and critic network ϕ;
  if $l > 1$ and $|J_t(\theta) - J_{t-1}((\theta))| + |L_t(\phi) - L_{t-1}((\phi))| < \epsilon_1$ **then**
    Go to **return**
  end if
end for
return θ, ψ and ϕ;

fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), out-of-view (OV), background clutters (BC), and low resolution (LR). We followed the standard evaluation metrics on these benchmarks. We used the one-pass evaluation (OPE) with the distance precision metric and overlap success plots metrics, where each tracker was initialized with the ground truth location until the end of each sequence. Specifically, the overlap success rate measures the overlap between predicted bounding boxes and ground truth bounding boxes, and the distance precision metric is the percentage of frames where the estimated location center error from the ground truth is smaller than a given distance threshold. In our experiments, we set the threshold distance as 20 pixels for all trackers. The VOT-2016 dataset consists 60 challenging videos from a set of more than 300 videos. The performance in terms of both accuracy (overlap with the ground-truth) and robustness (failure rate) is evaluated in our experiments. Noting that on VOT-2016 dataset, a tracker is restarted by the ground-truth in the case of a failure.

4.2 Implementation Details

We implemented our tracker in Python using the Pytorch library. The implementation was conducted on a PC with an Intel Core i7 3.4GHz CPU with 24GB RAM and the deep neural networks were trained on GeForce GTX 1080 Ti GPU

with 11GB VRAM. In our settings, the proposed tracker runs about 10 frames per second on these two benchmarks [48, 49].

Prediction Network: The prediction network has three convolutional layers which are initialized by the VGG-M [8] network which was pretrained on ImageNet [15]. The next two fully connected layers has 512 and 100 output units with ReLU activations. The output fully connected layer has 4 output units combined with the *tanh* activation.

Actor-Critic Network: The actor network has two fully connected layers of 100 and 4 output units with the $ReLU$ activation. The critic network is similar to the actor network but the final layer has only one output unit. The current and candidate features are concatenated as the input to these two networks. We use the Adam optimizer [27] with a learning rate of 0.0001 and a discount of β (set as 0.95) to train the actor-critic network. We trained our actor-critic network by using sequences which were randomly sampled from the VOT-2013, VOT-2014, and VOT-2015 [30] in which videos overlapping with OTB and Temple-Color were excluded. The maximal number of actions is set to 10 for each frame and the starting frame for each episode is randomly selected. The end operation is determined by the mean IoU ratio of the last 5 predicted bounding boxes compared to the ground truth bounding boxes of the total frames of one sequence. If the mean IoU is under 0.2 or at the end of a sequence, we terminate the episode and update the models. We trained the network for a total num of 50,000 episodes until convergence. On VOT-2016 dataset, we conducted experiments using ImageNet as the training set for our tacker. Since each object on the training set has only one frame (static image), we set γ as 0 in Eq. 12, and removed the action *stop & ignore*.

4.3 Results and Analysis

Quantitative Evaluation: We conducted quantitative evaluations on the OTB-2015 Dataset, Temple-Color Dataset and VOT-2016 Dataset.

OBT-2015 Dataset. We compared our approach with the state-of-the-art trackers including CREST [43], ADNet [52], MDNet [36], HCFT [32], SINT [45], DeepSRDCF [12], and HDT [39]. Fig. 4 shows the performance of different trackers in terms of precision and success rate based on center location error and overlap ratio on OTB-2015. We also evaluated the performance of different tracking methods and the processing speed (fps) on OTB-2015 dataset. Overall, our tracker performs favorably on both the precision and the success rate, meanwhile runs at 10.2 fps which is 5 times faster than the state-of-the-art tracker MDNet (2.1 fps in Pytorch implementation). One variant of our tracker with only two action types shown later runs even faster with an acceptable trade-off of accuracy.

We also analyzed the performance of our tracker for three different challenge attributes labeled for each sequence including fast motion, deformation, scale variations. We compute the OPE on the distance precision metric under 8 main video attributes. As shown in Fig. 5, our tracker shows competitive results on all the attributes. Specifically, the effectiveness in deformation attributes to the prediction network update according to the policy to capture target appearance

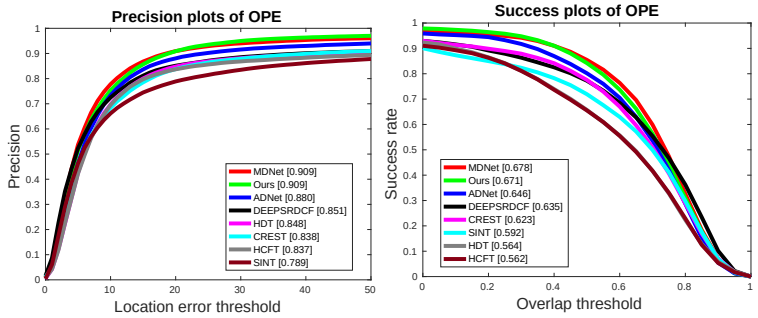


Fig. 4: The precision and success rate over all sequences by using one-pass evaluation on the OTB-2015 Dataset [49]. The legend includes the area-under-the-curve score and the average distance precision score at 20 pixels for each tracker

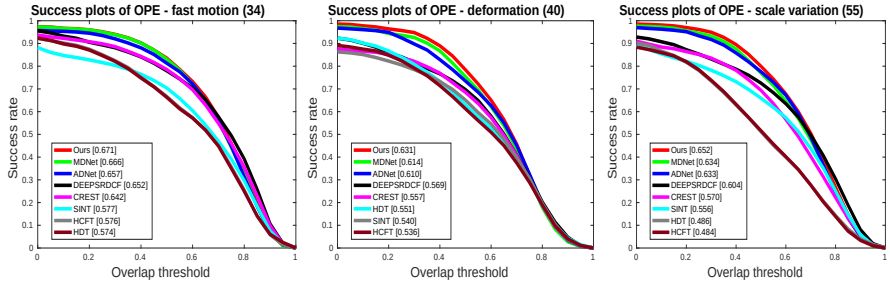


Fig. 5: The success plots over three tracking challenges, including fast motion, deformation, scale variations, for all the compared trackers on OTB-2015

changes. For scale variation, our tracker still performs well which demonstrates that our prediction network is robust to the scale change of the target object. Our tracker performs better on all three challenges than ADNet [52], which is also a deep reinforcement learning based tracker. The main reason is that our prediction network can be adjusted according to the action learned by the policy network. Meanwhile, the action *stop* & *ignore* and *stop* & *update* can guide our tracker whether to update the target feature, which avoids inadequate model update in long-term tracking. We have also obtained similar performance in fast motion, where MDNet [36] and our tracker both benefit from the convolutional features and the re-detection process. However, the percentage of the frames using re-detection to the total frames of MDNet [36] is high, resulting in more computation.

Temple-Color Dataset. We evaluate our approach on the Temple-Color dataset containing 128 videos. Fig. 7 shows the performance of different trackers in terms of precision and success rate based on center location error and overlap ratio. The C-COT tracker [13] and MEEM [54] reach the average distance precision score of 0.781 and 0.706. Our approach improves by a significant mar-

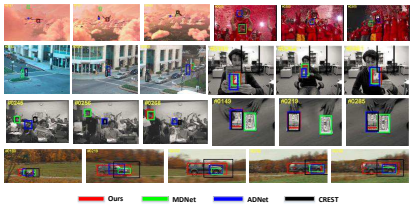


Fig. 6: Qualitative evaluation of our tracker, MDNet [36], ADNet [52] and CREST [43] on 7 challenging sequences

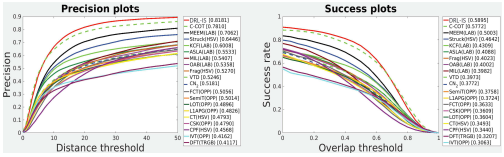


Fig. 7: The precision and success plots over all sequences by using one-pass evaluation on the Temple-Color Dataset. The legend contains the average distance precision score and the area-under-the-curve score for each tracker

Table 1: Comparison with state-of-the-art methods in terms of robustness and accuracy ranking on the VOT-2016 dataset(the lower the better)

Baseline	MDNet_N	DeepSRDCF	Staple	MLDF	SSAT	TCNN	C-COT	DRL-IS
Robustness	5.75	5.92	5.70	4.23	4.60	4.18	2.92	**2.70**
Accuracy	4.63	4.88	4.23	6.17	**3.42**	4.22	4.85	3.60

gin, achieving a score of 0.818. In the success plot in Fig. 7, our method also achieves a notable absolute gain of 1.2% in area-under-the-curve score compared to state-of-the-art method C-COT.

VOT-2016 Dataset. Table 3 shows the comparison of our approach with the top 5 competing trackers in the VOT-2016 challenge. As shown in Table 1, we obtain competitive accuracy and robustness ranking with state-of-the-art methods on the VOT-2016 Dataset. Our method achieves favorable results in terms of accuracy while keeping a low failure rate, which attributes to the decision making on motion estimation and tracking status guided by reinforcement learning. Noting that MDNet_N is a variation of MDNet, which does not pre-train CNNs with other tracking datasets. MDNet_N is also initialized using the ImageNet like our method. Our DRL-IS improves the performance of MDNet_N by a significant margin, which shows that our tracker has good generality without using the tracking sequences as training data.

Qualitative Evaluation: Fig. 6 shows qualitative comparisons of top performing visual tracking methods including MDNet [36], ADNet [52], CREST [43] and our method on 7 challenging sequences. Our tracker performs well against the compared these methods in all sequences. Moreover, none of the other methods is able to track targets for the *CarScale* sequence whereas our tracker successfully locates the target as well as estimates the scale changes. There are two reasons: 1) Our method accounts for the appearance changes caused by deformation and background clutters (*Bird*1, *Soccer* and *Freeman*4) by adjusting the bounding box of the object iteratively; 2) The feature of objects and the models are updated adaptively with deep reinforcement learning to account for appearance variations.

Table 2: The comparisons of different ablation variants of DRL-IS over the distance precision and overlap success plots on the OTB-2015 dataset

Variants	Shift (22 fps)	Shift+IS (15 fps)	DRL-IS (10.2 fps)
Prec.(20px)	0.822	0.887	0.909
IOU(AUC)	0.593	0.651	0.671

Ablation Study of Different Components: To show the impacts of different components of our tracker, we developed three variants of our tracker by integrating the prediction network with different types of policies combination and evaluated them using OTB-2015. These three variants are: 1) "Shift" is a baseline tracker which contains only one module based on the pre-trained prediction network; 2) "Shift + IS" is a pre-trained prediction model which was guided with only two action types: *continue* and *stop* & *update*; and 3) "DRL-IS" is our final model which was guided with full action types: *continue*, *restart*, *stop* & *ignore* and *stop* & *update*. Table 2 shows the distance precision and overlap success plots of these variations on the OTB-2015 dataset. The "Shift" tracker can only obtain the one-step shift based on deep convolutional features, which dose not perform well because the model is not updated during tracking and may fail when the target object changes fast. The "Shift + IS" tracker enables iterative shift and updates the model according to the policy learned by the actor network, which outperforms the baseline tracker by 6.5% and 5.7% in terms of the precision and overlap success, respectively. Moreover, "DRL-IS" incorporates all actions with the prediction network and achieves 8.7% and 2.2% performance gains of in terms of the precision over the "Shift" and "Shift + IS" variations, respectively.

5 Conclusion

In this paper, we have proposed a DRL-IS method for visual tracking, which has demonstrated reinforcement learning is an effective way to model the tough decision making process for tracking, *i.e.*, performing motion estimation and changing tracking status at the same time. The new iterative shift by deep nets locates targets efficiently than online classification and copes well with the cases that deformation or motion blur present in video. Extensive experiments on 3 public datasets have validated the advantages on tracking robustness and efficiency of the proposed method.

Acknowledgements

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 61672306, Grant U1713214, Grant 61572271, and in part by the Shenzhen Fundamental Research Fund (Subject Arrangement) under Grant JCYJ20170412170602564.

References

1. Ali, N.H., Hassan, G.M.: Kalman filter tracking. IJCA (9) (2014)
2. Ammar, H.B., Eaton, E., Ruvolo, P., Taylor, M.: Online multi-task learning for policy gradient methods. In: ICML. pp. 1206–1214 (2014)
3. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR. pp. 983–990 (2009)
4. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust l1 tracker using accelerated proximal gradient approach. In: CVPR. pp. 1830–1837 (2012)
5. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: CVPR. pp. 2544–2550 (2010)
6. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: ICCV. pp. 2488–2496 (2015)
7. Cao, Q., Lin, L., Shi, Y., Liang, X., Li, G.: Attention-aware face hallucination via deep reinforcement learning. In: CVPR. pp. 690–698 (2017)
8. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv (2014)
9. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: CVPR. pp. 142–149 (2000)
10. Cui, Z., Xiao, S., Feng, J., Yan, S.: Recurrently target-attending tracking. In: CVPR. pp. 1449–1458 (2016)
11. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. TPAMI (8), 1561–1575 (2017)
12. Danelljan, M., Hager, G., Shahbaz Khan, F., Felsberg, M.: Learning spatially regularized correlation filters for visual tracking. In: ICCV. pp. 4310–4318 (2015)
13. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: ECCV. pp. 472–488. Springer (2016)
14. Decarlo, D., Metaxas, D.: Optical flow constraints on deformable models with applications to face tracking. IJCV (2), 99–127 (2000)
15. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
16. Gao, J., Ling, H., Hu, W., Xing, J.: Transfer learning based visual tracking with gaussian processes regression. In: ECCV. pp. 188–203 (2014)
17. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. pp. 580–587 (2014)
18. Gordon, D., Farhadi, A., Fox, D.: Re3 : Real-time recurrent regression networks for object tracking. CoRR (2017)
19. Hager, G.D., Belhumeur, P.N.: Efficient region tracking with parametric models of geometry and illumination. TPAMI (10), 1025–1039 (1998)
20. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI (3), 583–596 (2015)
21. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: ICCV. pp. 105–114 (2017)
22. Isard, M., Blake, A.: Condensation–conditional density propagation for visual tracking,(1998). IJCV pp. 5–28
23. Jie, Z., Liang, X., Feng, J., Jin, X., Lu, W., Yan, S.: Tree-structured reinforcement learning for sequential object localization. In: NIPS. pp. 127–135 (2016)
24. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. TPAMI (7), 1409–1422 (2012)

25. Kamalapurkar, R., Andrews, L., Walters, P., Dixon, W.E.: Model-based reinforcement learning for infinite-horizon approximate optimal tracking. TNNLS (3), 753–758 (2017)
26. Karayev, S., Baumgartner, T., Fritz, M., Darrell, T.: Timely object recognition. In: NIPS. pp. 899–907 (2012)
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR (2014)
28. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: NIPS. pp. 1008–1014 (2000)
29. Kong, X., Xin, B., Wang, Y., Hua, G.: Collaborative deep reinforcement learning for joint object search. In: CVPR (2017)
30. Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., Čehovin, L.: A novel performance evaluation methodology for single-target trackers. TPAMI (11), 2137–2155 (2016)
31. Liang, P., Blasch, E., Ling, H.: Encoding color information for visual tracking: Algorithms and benchmark. TIP (12), 5630–5644 (2015)
32. Ma, C., Huang, J.B., Yang, X., Yang, M.H.: Hierarchical convolutional features for visual tracking. In: ICCV. pp. 3074–3082 (2015)
33. Mathe, S., Pirinen, A., Sminchisescu, C.: Reinforcement learning for visual object detection. In: CVPR. pp. 2894–2902 (2016)
34. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature (7540), 529–533 (2015)
35. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. CoRR (2015)
36. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: CVPR. pp. 4293–4302 (2016)
37. O'Donoghue, B., Munos, R., Kavukcuoglu, K., Mnih, V.: Pgq: Combining policy gradient and q-learning. arXiv preprint arXiv:1611.01626 (2016)
38. Okuma, K., Taleghani, A., De Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: ECCV. pp. 28–39 (2004)
39. Qi, Y., Zhang, S., Qin, L., Yao, H., Huang, Q., Lim, J., Yang, M.H.: Hedged deep tracking. In: CVPR. pp. 4303–4311 (2016)
40. Rao, Y., Lu, J., Zhou, J.: Attention-aware deep reinforcement learning for video face recognition. In: ICCV. pp. 3931–3940 (2017)
41. Smeulders, A.W., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M.: Visual tracking: An experimental survey. TPAMI (7), 1442–1468 (2014)
42. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R.W.H., Yang, M.: CREST: convolutional residual learning for visual tracking. CoRR (2017)
43. Song, Y., Ma, C., Gong, L., Zhang, J., Lau, R.W.H., Yang, M.H.: Crest: Convolutional residual learning for visual tracking. In: ICCV. pp. 2555–2564 (2017)
44. Supancic, III, J., Ramanan, D.: Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In: ICCV. pp. 322–331 (2017)
45. Tao, R., Gavves, E., Smeulders, A.W.: Siamese instance search for tracking. In: CVPR. pp. 1420–1429 (2016)
46. Wang, L., Ouyang, W., Wang, X., Lu, H.: Stct: Sequentially training convolutional networks for visual tracking. In: CVPR. pp. 1373–1381 (2016)
47. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: NIPS. pp. 809–817 (2013)
48. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. In: CVPR. pp. 2411–2418 (2013)

49. Wu, Y., Lim, J., Yang, M.H.: Object tracking benchmark. TPAMI (9), 1834–1848 (2015)
50. Yang, B., Nevatia, R.: Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In: CVPR. pp. 1918–1925 (2012)
51. Yang, H., Shao, L., Zheng, F., Wang, L., Song, Z.: Recent advances and trends in visual tracking: A review. Neurocomputing (18), 3823–3831 (2011)
52. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR. pp. 2711–2720 (2017)
53. Zhang, D., Maei, H., Wang, X., Wang, Y.F.: Deep reinforcement learning for visual object tracking in videos. arXiv preprint arXiv:1701.08936 (2017)
54. Zhang, J., Ma, S., Sclaroff, S.: Meem: robust tracking via multiple experts using entropy minimization. In: ECCV. pp. 188–203 (2014)