STL_Siam: Real-time Visual Tracking based on reinforcement guided network

Huang Shijia¹,², Wang Luping¹,²

¹School of Electronics and Communication Engineering, Sun Yat-sen University, China Guangzhou, China
²School of Electronics and Communication Engineering, Sun Yat-sen University, China Guangzhou, China
ahuangshj39@mail2.sysu.edu.cn, bwanglp27@mail.sysu.edu.cn

Abstract—In recent years, deep visual tracking algorithms based on Siamese have made great breakthrough in both speed and accuracy. However, due to the dependence of Siamese network on the target template, these trackers are prone to drift or even fail to track in a complex tracking environment. In this work, we navigate the Siamese network with a STLNet model and propose the STL_Siam method. The STLNet, trained offline with the dataset enhanced by ROMIX method, is introduced into the SiamGrad algorithm to infer the movement of the target during online tracking, and guides the SiamGrad network to track. In order to evaluate the reliability of the proposed algorithm, we conducted experiments on the OTB2013 and VOT2016 two benchmarks. The algorithm achieved excellent performance, and the performance of the sequences with complex obstruction interference has been slightly improved. Experiments suggest that the proposed architecture reaches 40+fps and gets 0.865 precision on OTB2013, which is higher than results of the ACT. Meanwhile, compared with the baseline algorithm ACT, the A, R, and EAO of the proposed approach is increased by 5.6%, 1%, and 0.3% respectively on VOT2016 respectively.

1. INTRODUCTION
Visual object tracking is one of research focuses in the field of computer vision, and it has broad application prospects in industry, military, and intelligent surveillance. Given a rectangular bounding box of the target in the first frame of the video, the trackers could continue to track the object in following frames. Traditional tracking algorithms are essentially divided into two categories: One is based on matched filters, including NCC[1,2,3] and KCF[4]. They aim to train a target detector with target patches as positive samples, and the surrounding patches as the negative and then detector detects whether the predicted position of the next frame is the target. Meanwhile, the reliable detection results are also applied to updating the detector. However, due to the lack of self-adaptation for target template initialized in the first frame, the scale change of the target in the tracking sequence would cause the target drift during tracking, resulting in tracking failure. Further when the environment owns a tremendous change, the detector will learn more background information and disturb the trackers’ decision. The other methods depend on machine learning, such as TLD[5] and Struck[6]. Via a certain trained model to endow trackers with self-learning and judgment capabilities, the object can change adaptively and be tracked stably. They also can be subdivide into two types containing integrated learning methods and discriminative methods: the integrated learning methods are to model the target in the current frame, and find the most similar patches as the predicted position in next frame; while the discriminative methods regard the tracking
problem as a two-classification problem, it follows that the tracking models consist of feature extraction and classifier. Specifically, they extract the feature from images and then train the classifier with positive and negative samples to discover the optimal target location.

Since the powerful feature description through convolutional neural networks (CNNs) with mass of labeled data, the deep algorithms make great progresses in computer visions. Visual object tracking only provides the bounding box of the first frame as training data. In this case, it is very difficult to train a deep model from scratch for the current target at the beginning of tracking. In this case, the end-to-end deep learning tracking algorithms have achieved great success. Siamese network tracking algorithms are a pioneering model of end-to-end tracking networks owing to their speed and accuracy. These methods primarily determine the new position of the target by calculating the correlation between the candidate area of the subsequent frame and the target area with the similarity measurement. In 2016, Bertinetto[7] proposes the SiamFC model, who first applies the Siamese network to the deep model. SiamFC could reach 86 frames per second even using 3 scale estimations, fully meets the real-time requirements and breaks through the limits of deep learning tracking algorithms with its high speed and robustness. The SiamFC network cannot adjust the shape of the tracking target. But the regression prediction of the target is as important as the positioning of the tracking target for tracking. Bo Li [8] proposes SiamRPN algorithm to combine candidate domain generation network RPN into Siamese network and abandons multi-scales strategy and online fine-tuning. SiamRPN model converts the original similarity calculation problem into a regression and classification problem and predicts the object quicker and more robustly. In [9], Qing Guo et al. solves the imbalance of positive and negative sample blocks as well as the lack of the richness of sample blocks. That is, more attention is paid to the problem of input data, and the tracking performance is improved through data enhancement methods. Wang Qiang et al.[10] proposes an approach which adds a mask branch to SiamFC to solve the problems of object tracking and object segmentation. Besides the mask branch makes the detection result more accurate. In [11], SiamRPN++ uses the deeper network as the backbone, and utilizes more innovative methods including multi-layer feature fusion and multi-layer RPN scheme, achieved the current best accuracy on multiple benchmark datasets. However, there are still some deficiencies in the specific scenarios. For example, when the target is occluded for a long time or the target is out of view, it will directly lead to a false model update, and finally the tracking fails. Provided that reduced the learning rate, the algorithm model might not adapt to the model change. When the target reappears, how to retrieve the target quickly and efficiently is still a difficult problem.

2. RELATED WORK

2.1. ROIMIX

ROIMIX[12] is a data enhancement approach, which first coalesces multiple images to characterize the interaction between them. By means of mixing candidate regions between multiple images, ROIMIX simulates the situation of object overlap, occlusion and blurring, and could improve the generalization ability. While tracking is a sequential behavior and also suffers from deformation, occlusion and etc. under a complete environment. In this case, as a robust tracker, it needs to be able to distinguish these targets that encounter interference. Therefore, we train the model SiamGrad offline with ROMIX to improve the robustness of the model.

2.2. Deep Reinforcement learning

Reinforcement learning[13,14,15] is a machine learning algorithm that makes decisions and actions based on current conditions to achieve an expected goal. Usually use Markov decision process[16] description. Markov decision process refers to the random process in which the agent observes Markov and makes decisions at different times. It has no aftereffect like Markov stochastic process, that is, it is assumed that the next state of the target is only related to the current state:

$$p(s_{i+1}|s_i) = p(s_{i+1}|s_1, s_2, \cdots, s_i) \quad (1)$$
Reinforcement learning is mainly based on the reward mechanism, thus the Markov decision process introduces the reward value $R$ in its description, which can be described by a four-tuple:

$$M = \langle S, A, P_{s,a}, R \rangle$$

where in $s \in S$, $S$ represents the set of all environmental states, $s$ is a specific state; $A$ denotes the action set predefined by the agent, and $a$ represents a specific action in $a \in A$. $P(s, a)$ indicates the probability of state transition if the agent makes action. $R(s, a)$ represents the instant reward that the agent gets after an action.

As shown in Figure 1, reinforcement learning aims to maximize accumulated rewards through continuous trial and error learning, and constantly adjusts its own action strategy, so that the agent reaches the state of completing the target task.

However, traditional reinforcement learning is limited to small action spaces and sample spaces, and is generally discrete. In reality, tasks are more complex and often require a large state space and continuous action space. Thus, some scholars propose deep reinforcement learning, which attach deep learning to reinforcement learning, and learn control strategies directly from high-dimensional raw data. It integrates strong understanding of perception of deep learning and realizes end-to-end learning, making reinforcement learning technology truly practical and able to solve complex problems in real-world scenarios. For tracking algorithms based on Siamese network, it mainly learns a matching function with a large number of sample pairs and achieve the tracking task. In the tracking process, it often ignores the information between frames, and uses inefficient templates to find the optimal target location. Due to these shortcomings, some scholars gradually introduce deep reinforcement learning into visual tracking. CHEN[17][16] proposed an adaptive tracking method of in-depth feature flow cascades, which adopts a Q-Learning[18] module to make decisions, and could select the earlier feature flow to output. Otherwise, the tracker would walk through the whole network to search the target patch. Yun et al.[19] applies the reward mechanism in reinforcement learning to endow tracker with the ability of autonomously determining how to "close" to the target object to achieve target tracking. Chen et al.[20] propose an action-critic network, which could directly obtain the action of the tracked object via reinforcement learning. In this paper, we propose a reinforcement learning model STLNet based on the Resnet50[21] network. The model regards the tracking problem as a sequential decision problem, and finally outputs the relevant positioning information of the target when tracking, and guides the SiamGrad network to track. The STLNet training is mainly based on the DDPG[22] strategy, which can stably extract the relevant information of the target, and finally output the relative movement information of the target in the tracking process.

3. PROPOSED ALGORITHM

The algorithm framework proposed in this paper is shown in Figure 3. In order to improve the robustness of the tracker in complex environments, we propose the STL_Siam algorithm including STLNet and SiamGrad module. The STLNet module primarily learns the action of target via the DDPG reinforcement learning method, and finally outputs the movement of the tracking target center. For the offline training of STLNet model, we apply the ROMIX strategy to simulate the complex scene like occlusion and etc., making the model more robust. After received the prediction action from STLNet module, the SiamGrad module calculates the similarity between the candidate patch and template, and determines that the position with the highest similarity score as the tracked object. During the tracking process, the target template will be updated with gradient information to ensure the reliability of the template.
3.1. Guided Network STLNet based on DDPG method

The task of visual tracking is to locate the target in subsequent frames given the bounding box of target in the initial frame of a video sequence. That is, tracking is a sequential decision process as the reinforcement learning. Hence, we adopt the DDPG to achieve the target location task. First, we define the state $s$ as the image patch corresponding to the bounding box $[x, y, w, h]$ where $(x, y)$ is the center position of the patch, and $w$ and $h$ represent the width and height. And $a = [\Delta x, \Delta y, \Delta s]$ describes the relative motion of the tracking object, which respectively represents the target changes in the horizontal direction, vertical direction, and scale. At the same time, considering the time continuity of the tracking process, we constrain action $a: -1 \leq \Delta x \leq 1, -1 \leq \Delta y \leq 1, -0.05 \leq \Delta s \leq 0.05$ and get the new location $[x', y', w', h']$ by

$$
x' = x + \Delta x \times w
\hfill (2)
$$
$$
y' = y + \Delta y \times h
\hfill (2)
$$
$$
w' = w + \Delta s \times w
\hfill (2)
$$
$$
h' = h + \Delta s \times h
\hfill (2)
$$

And eventually calculates the new state $s' = f(s, a)$. The reward function $r(s, a)$ represents the improvement in accuracy when state transition occurs. Taking into account the important influencing factors of the target regression including overlap and center point distance, the algorithm introduces a reward mechanism based on the overlap rate of the predicted bounding box $b$ and the ground truth bounding box $G$.

$$
r(s, a) = \begin{cases} 1 & \text{if } DIOU(b, G) > 0.7 \\ -1 & \text{else} \end{cases} \hfill (3)
$$

where $DIOU$ is the Intersection over Union of distance, whose $b, b' \in \mathbb{R}$ represent the center points of the predicted frame and the real frame respectively, $\rho$ represents the Euclidean distance between the two center points, $c$ represents the diagonal distance of the smallest closed area that can contain both the prediction box and the real box, $IOU$ refers to the intersection ratio of two bounding boxes.

$$
DIOU = IOU - \frac{\rho^2(b, b')} {c^2} \hfill (4)
$$

In this work, we input state into the STLNet model to infer the best action $a$, that is, $a = u(s | o_u)$, where $u(.)$ represents the deep network of STLNet with parameter $o_u$ (details in Section A.I). In detail, after experimental verification, we finally adopt the dual bounding box scheme to train the STLNet model (here, two target size bounding boxes which their center position is the same would be input to be model).

### 3.1.1. Offline training of STLNet model

Network structure: The ResNet50 network is used as the backbone of STLNet model. As shown in Figure 2, after the input image block through ResNet50, we get 2 output nodes with a size of $1 \times 512$. And with the ReLU operation, the output size will be $1 \times 1024$ nodes and eventually generate a three-dimensional output after passing through 2 continuous fully connected layers.

Training: The STLNet model is trained by DDPG method, and is iteratively update the STLNet model parameters based on the sample pairs collected via reinforcement learning. Given $N$ groups $s_i, a_i, r_i, s'_i$, value function $Q(s, a)$ in the STLNet network is learned by Bellman’s equation. Hence the learning process can be realized by minimizing the loss function based on the target network $u(.)$ and $Q'$ function.

$$
L = \frac{1} {N} \sum \left( y_i - Q(s_i, a_i | \theta^Q) \right)^2 \hfill (5)
$$

where, $y_i = r_i + \gamma Q'(s'_i | \theta^u)$. Then, the chain rule is applied to update the STLNet model parameters.

$$
\nabla \theta_Q \approx \frac{1} {N} \sum \nabla Q(s_i, a_i | \theta^Q) |_{s=s_i, a=u(s_i)} \nabla \theta_u u(s_i | \theta^u) |_{s=s_i} \hfill (6)
$$

In the training process, we randomly select a training sequence segment from the training data set $[F_k, F_{k+1}, \ldots, F_{k+T}]$ and the corresponding ground truth $[G_k, G_{k+1}, \ldots, G_{k+T}]$ and then obtain the training pair $(s_t, a_t, r_t, s'_t)$ via STLNet model in $t$ frame, see the specific implementation on algorithm 1.

Algorithm 1: the offline training of STLNet model
Input: Training sequences [F] and related ground truth [G]
Output: the predicted action and trained STLNet model parameters

Training:
- Initialize the function \( Q(s, a) \) and STLNet network \( \mu(s|\theta_\mu) \) randomly, which their parameter weight is \( \theta_Q, \theta_\mu \) respectively.
- Initialize the target network \( Q' \) and \( u' \) and their parameter weights is as same as \( Q(s, a) \) and STLNet network \( \mu(s|\theta_\mu) \) : \( \theta_Q' \leftarrow \theta_Q, \theta_u' \leftarrow \theta_u \).
- Initialize the set of experience playback \( R \)

Repeat:
- Randomly select a training sequence segment \([F_k, F_{k+1}, \ldots, F_{k+T}]\) and related ground truth \([G_k, G_{k+1}, \ldots, G_{k+T}]\) according to the initial observation state \( s_k \) obtained from the \( F_k \) and \( G_k \), generate batches samples \((s_i, a_{i,\text{true}})\) and STLNet is trained in an iterative cycle.
- for each \( t=2, T+1, \) do:
  a. Obtain state \( s_t \) via \( s_{t-1} \) and \( F_{t-1+k} \)
  b. Select action \( a_t = \mu(s_t|\theta_u) \)
     according to current strategy and exploration probability \( \theta \)
  c. When the model output \( a_t \) , it gets the predicted location via formula (2), and obtains the reward \( r_t \), states \( s_t' \) via formula (3).
  d. Save \((s_t, a_t, r_t, s_t')\) to collection \( R \)
end for
- Randomly select \( N \) groups \((s_i, a_i, r_i, s_i')\) from \( R \) as a batch.
- Minimize loss based on formula (5) and update \( Q(s, a) \)
- Update \( \mu(s|\theta_\mu) \) based on formula (6) and policy gradient strategy
- Update target network:
  \[ \theta_Q' \leftarrow \tau \theta_Q + (1 - \tau) \theta_Q' \]
  \[ \theta_u' \leftarrow \tau \theta_\mu + (1 - \tau) \theta_u' \]
Until Reward stabilizes

Figure 2 the network structure of STLNet model
3.1.2. ROMIX method on the STLNet model:
ROMIX can simulate the situation for object occlusion, blur, jitter, etc. and thus improve the detection ability of the dense environment on the implicit models. In order to improve the anti-occlusion ability of the tracker, we adopt the data enhance operation via ROMIX method for STLnet training dataset. Details are showed as follows. Let \( x \) be the candidate patch to be input to the network, and \( k \) is the candidate block cropped from others. According to the Beta distribution, the mixed image block \( x' \) is

\[
\begin{align*}
\lambda &= B(a, a), \\
\lambda' &= \max(\lambda, 1 - \lambda)
\end{align*}
\]

Due to the huge action space, it is extremely difficult for the agent to train an efficient model by using a random search strategy. Besides, it would produce extremely unbalanced training samples, making the DDPG method inefficient. To solve this problem, we initialize the STLNet network from the supervision information of the initial frame to update the current environment. That is, the STLNet model is fine-tuned through adaptive moment estimation. In addition, if only adopt the actions generated by STLNet as sample training, it cannot completely solve the imbalance between positive and negative samples, because there are many unpredictable challenges that lead to tracking drift. We use probabilistic expert decision guidance to guide the exploration in reinforcement learning. It would replace the actions output by the STLNet network with a certain probability \( \theta \), while the probability \( \theta \) gradually decreases during training.

3.2. STL_Siam algorithm based on SiamGrad
STL_Siam model consists of STLNet module and SiamGrad module. STLNet module primarily applies the prior information to infer the target action in next frame and then guides the SiamGrad module to locate the target. While the SiamGrad module aims to search the target and owns a template update branch to generate the more robust template, detailed in Section B.I and B.II.

3.2.1. Basic Tracker of SiamGrad model
The SiamGrad module mainly regards SiamFC as the basic tracker. By comparing the similarity of the template and candidate regions, the SiamFC adopts the position with the largest score in the score map as the predicted position of the target.

\[
S = g(t_x, \beta) = t_x * t_z
\]

where \( x \) describes a search region and \( z \) denotes the target patch. \( t_x = \varphi(x), t_z = \varphi(z) \) respectively denotes extracted feature of searched region and template. * represents the cross-correlation convolution and \( S \) is the score map.

3.2.2. Template Update Strategy based on gradient
In order to enhance the discriminative ability of template \( \beta \) during online tracking, we adopt a gradient-based template update method[23]:
\[ \beta_{grad} = N(Z, X, \alpha) \]  

where \( Z \) represents template, \( X \) is the search patch and \( \alpha \) is the parameters that extract the features of template \( Z \), but also the background information in \( X \) through the gradient. And the implementation describes as following:

a. Given image pair \((X, Z)\), the template features \( t_2(Z) \) is obtained from two-convolutional layer. Meanwhile, it would enter the subnet \( N_1 \) to get the template \( \beta \):

\[ \beta = N_1(t_2(Z), \alpha_1) \]

(12)

where \( \alpha_1 \) are parameters of \( N_1 \). After obtaining the initial template \( \beta \), use formula (9) to calculate the initial score map \( S \):

b. We calculate the logical loss with the initial score map \( S \) and label:

\[ \text{loss} = l(S, Y) \]

(13)

Then the loss is used to calculate the gradient of \( t_2(Z) \) and add it to the \( t_2(Z) \), and we receive the updated template feature from subnet \( N_2 \):

\[ h_2(Z) = t_2(Z) + N_2 \left( \frac{\partial \text{loss}}{\partial t_2(Z)}, \alpha_2 \right) \]

(14)

where \( \alpha_2 \) are parameters of \( N_2 \). In addition, the gradient in connection with \( N_1 \) is utilized as the input of the sub-net \( N_2 \) to obtain the final loss.

c. Eventually, we send the \( h_2(Z) \) to subnet \( N_1 \) and calculate the optimal template \( \beta^* \) and final score map \( S^* \):

\[ \beta^* = N_1(h_2(Z), \alpha_1) \]

(15)

\[ S^* = \beta^* * t_2(X) \]

(16)

Meanwhile, in order to make the model have a good sense of the target, we minimize the loss of \( S^* \) to update the branch for offline training: \( \text{arg min}_{a} \sum l(S^*, Y) \)

3.3. Online Tracking

After offline training of STLNet and template generation network, we directly initialize the model to track.

- Initialization: we initialize the STLNet model and obtain the best template \( \beta^* \) at the first frame. Specifically, we select \( M \) candidate target boxes around \( g_t \) and calculate the corresponding action, input them to STLNet for initialization and use the Adam method to minimize \( L_2 \) loss:

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

(17)

- Online tracking: we extract image patches according to previous frame, and input them into STLNet to obtain the target transformation. The SiamGrad tracker would analyze the target change from STLNet module and locate the target.

- Template update online: It iteratively update the template \( \beta^* \) when obtains a reliable sample \((X, Y)\) via formula (12-16) (That is to replace \( t_2(Z), (X, Y) \) with \( h_2(Z), (X_t, Y_t) \))

4. EXPERIMENT

Our algorithm is mainly based on Tensorflow framework, runs on 64-bit intel i7 3.2GHz CPU and 6GB GeForce GTX 1060 and reaches 40+fps.

4.1. Analysis on STLNet model

This paper utilizes ResNet50 as the backbone network of STLNet model. In this section, we directly use the trained STLNet as the backbone of the tracker to locate the target. First, the STLNet model derive the relative movement of the target and then use the offline trained Critic network as a two-class network to judge whether the predicted result is reliable. Concluded from the experiments on OTB2013 and VOT2016 datasets, the method based on ResNet50 could quickly minimize the loss when initializing the model (it only takes a few epochs to reach the threshold), and the tracker is more stable than that based on the VGG model. Table 1 shows the experimental results of the algorithms on the OTB2013 dataset. It
shows that the robustness of STLNet based on ResNet50 will be stronger than that based on VGG when track on those sequences with attributes of OCC, MB, and FM, increasing 1.8% accuracy in the average.

**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) ON OTB2013.

| Method | IV   | SV   | OCC  | DEF  | MB   | FM   | IPR  | OVR  | IPD  | LR   | LR   | AV   |
|--------|------|------|------|------|------|------|------|------|------|------|------|------|
| ACC_NET| 0.815| 0.845| 0.839| 0.803| 0.854| 0.836| 0.862| 0.873| 0.61 | 0.923| 0.929| 0.848|
| ACT   | 0.836| 0.843| 0.799| 0.825| 0.810| 0.805| 0.867| 0.837| 0.788| 0.881| 0.887| 0.833|
| SiamRPN| 0.845| 0.843| 0.753| 0.825| 0.831| 0.795| 0.857| 0.854| 0.738| 0.893| 0.847| 0.823|
| GradNet | 0.844| 0.843| 0.838| 0.795| 0.894| 0.820| 0.866| 0.828| 0.789| 0.922| 0.929| 0.844|
| ECO-HC | 0.929| 0.929| 0.848| 0.806| 0.615| 0.819| 0.8  | 0.834| 0.818| 0.859| 0.895| 0.822|
| SiamFC  | 0.736| 0.740| 0.733| 0.691| 0.736| 0.742| 0.743| 0.758| 0.673| 0.692| 0.516| 0.731|
| BCF     | 0.719| 0.933| 0.639| 0.617| 0.615| 0.620| 0.701| 0.617| 0.961| 0.712| 0.349| 0.634|
| RES     | 0.899| 0.928| 0.914| 0.747| 0.821| 0.919| 0.824| 0.730| 0.790| 0.792| 0.167| 0.766|

4.2. *Experimental analysis on STL_Siam algorithm*

We conduct the experiments on OTB2013 and VOT2016 and compares the proposed algorithm STL_Siam with other 6 trackers with real-time, including GradNet, ACT, ECO-HC[26], SiamRPN, SiamFC and KCF. Expect the KCF algorithm, the other 6 methods are based on CNN.

In the OTB2013 benchmark, we use both the precision and success rate to evaluate different trackers. Figure 4 shows the accuracy and success rates of these trackers in the OTB dataset. It can be observed that the accuracy of the STL_Siam model on OTB2013 reaches 0.865, which is 0.2%~0.4% higher than the GradNet and ACT algorithms for precision and success rates. In general, the STL_Siam model is competitiveness among them.

On the VOT2016 dataset, there are 60 short sequences with 6 different attributes and regard accuracy (A) and robustness (R) and expected average overlap (EAO) as evaluation criteria. In Table 2, it is observed that STL_Siam is 5.6%, 1%, and 0.3% higher than the baseline algorithm ACT on A, R, and EAO respectively. Compared with the trackers SRDCF and ECO-HC based on correlation filtering, it has great performance. Because the tracker based on correlation filter is susceptible to the bounding effect. While reinforcement could learn sufficient information, and it is not limited by the supervisor and thus STL_Siam method rises superior to others.

Figure 4 The precision and success plots of different trackers on the OTB2013
Figure 5 Some results of STL_Siam and other algorithms

Table 2 The accuracy (A), robustness (R) and expected average overlap (EAO) scores of different trackers on VOT2016

|        | Accuracy | Robustness | EAO  |
|--------|----------|------------|------|
| ACT    | 0.49     | 0.481      | 0.274|
| SiamRPN| 0.561    | 0.265      | 0.341|
| SiamFC | 0.532    | 0.469      | 0.249|
| ECO-HC | 0.543    | 0.309      | 0.322|
| SRDCF  | 0.529    | 0.326      | 0.276|
| GradNet| 0.517    | 0.375      | 0.267|
| Ours   | 0.546    | 0.491      | 0.277|

5. CONCLUSION

In this article, we propose a Siamese tracking network STL_Siam embedded with a reinforcement learning module to handle the tracking problems. The STLNet module adopts the ROMIX method to simulate the complex environment and thus enhances the model with powerful characterization capabilities. After that, it would send its inferred information for the SiamGrad tracking module to guide tracker. The algorithm has achieved excellent performance on the two benchmarks OTB100 and VOT2016. For scenes with serious interference such as obstructions, the tracking performance has been significantly improved compared with the other trackers. On VOT2016, its A, R, and EAO are increased by 5.6%, 1%, and 0.3% compared with the baseline algorithm ACT respectively. At the same time, the success rate and precision of STL_Siam on OTB100, are also higher than ACT, indicating the feasibility of the deep tracking algorithm network based on reinforcement learning.

REFERENCES

[1] HE Xiaojia, Research on fast gray image matching algorithm [D]. Hefei: HeFei University of Technology, 2012
[2] XIE Weida, ZHOU Yuheng, KOU Ruolan. An improved fast normalized cross correlation algorithm[J]. Journal of Tongji University: Natural Science, 2011, 39(8):1233–1237.

[3] SUN Bujiao, ZHOU Donghua. Fast matching method based on NCC [J]. Transducer and microsystem technologies, 2013, 40(1):118–125.

[4] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(3):583–596, 2015.

[5] Kalal Z, Mikolajczyk K, Matas J. Tracking-Learning-Detection. IEEE Trans Pattern Anal Mach Intell. 2012;34(7):1409-1422. doi:10.1109/TPAMI.2011.239

[6] Hare, Sam, A. Saffari, and P. H. S. Torr. "Struck: Structured output tracking with kernels." IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011 IEEE, 2011.

[7] Bertinetto L., Valmadre J., Henriques J. F., Vedaldi A., and Torr P. H. 2016. Fully-convolutional siamese networks for object tracking. In European conference on computer vision, 850–865. Springer.

[8] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In CVPR, 2018

[9] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. Learning dynamic siamese network for visual object tracking. In ICCV, 2017.

[10] Wang Q, Zhang L, Bertinetto L, et al. Fast Online Object Tracking and Segmentation: A Unifying Approach[J]. 2018.

[11] Li B, Wu W, Wang Q, et al. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks[J]. 2018.

[12] Lin W, Zhong J, Liu S, et al. RolMix: Proposal-Fusion among Multiple Images for Underwater Object Detection.[J]. arXiv: Computer Vision and Pattern Recognition, 2019.

[13] Caicedo, J. C. and Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In ICCV.

[14] Jie, Z., Liang, X., Feng, J., Jin, X., Lu, W. F., and Yan, S. (2016). Tree-structured reinforcement learning for sequential object localization. In NIPS.

[15] Kong, X., Xin, B., Wang, Y., and Hua, G. (2017). Collaborative deep reinforcement learning for joint object search. In CVPR.

[16] Whittle P, Puterman M L. Markov Decision Processes[J]. Journal of The Royal Statistical Society Series A-statistics in Society, 1994, 158(3).

[17] Chen Huang, Simon Lucey, Deva Ramanan. "Learning Policies for Adaptive Tracking with Deep Feature Cascades." ICCV

[18] Watkins C, Dayan P. Technical Note: Q-Learning[J]. Machine Learning, 1992, 8(3): 279-292.

[19] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, Jin Young Choi. "Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning.CVPR2017

[20] Boyu Chen, Dong Wang, Peixia Li, Huchuan Lu."Real-time Actor-Critic Tracking." ECCV (2018)

[21] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. 2016.

[22] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2015) Continuous Control with Deep Reinforcement Learning. arXiv preprint arXiv:1509.02971

[23] Li P, Chen B, Ouyang W, et al. GradNet: Gradient-Guided Network for Visual Object Tracking[C]. international conference on computer vision, 2019: 6162-6171.

[24] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9):1834–1848, 2015.

[25] Kristan, M., Leonardis, A., Jiri Matas, e.a.: The visual object tracking VOT2016 challenge results. In: ECCVW. (2016)

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