This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

NoisyOTNet: A Robust Real-Time Vehicle Tracking Model for Traffic Surveillance

Weiwei Xing, Yuxiang Yang*, Shunli Zhang, Qi Yu, Liqiang Wang

***Abstract*—With the rapid development of intelligent transportation, automated traffic surveillance is considered as an important component. In the field of traffic surveillance, it is particularly important to achieve robust and real-time tracking of vehicles in complex scenes. In this paper, a robust real-time vehicle tracking model named *NoisyOTNet* is proposed, which formulates tracking as reinforcement learning with parameter space noise. In this formulation, the exploration ability of the model is enhanced to improve the robustness of tracking. Specifically, we develop a new implementation for noisy network based on deep deterministic policy gradients (DDPGs) with parameter noise, which can better cope with the tracking task and directly predict the tracking result. To improve the tracking accuracy in complex conditions, e.g. fast motion and large deformation, this paper presents an adaptive update strategy that can exploit the vehicle spatial-temporal information based on Upper Confidence Bound (UCB) algorithm by exploiting. Moreover, as for the recovery of the lost target, a relocation algorithm based on incremental learning is developed. The results of extensive experiments demonstrate that the proposed NoisyOTNet can effectively track vehicles in complex scenes and achieve competitive performance compared to the state-of-the-art methods.**

***Index Terms*—Traffic surveillance, vehicle tracking, deep reinforcement learning, parameter space noise.**

I. INTRODUCTION

AUTOMATED traffic surveillance [1] is critical to intelligent transportation systems, which consist of multiple sub-tasks, such as detection, tracking, and recognition [2]–[5]. As a fundamental task of vehicle surveillance, an effective vehicle tracking algorithm can provide accurate vehicle position and tracking information for subsequent high-level semantic tasks. A vehicle tracking model consists of three components: an appearance module, a tracking module, and an update module [6], [7]. The appearance module is designed to represent the target by features. It is initialized at the beginning of tracking and is updated in the following frames according to different update strategies. The tracking module is used to locate the vehicle based on the appearance module. The update module updates the model based on changes of the target. The tracking process is to track the target with the above modules and output the target's position in each frame. A robust tracker can still achieve high accuracy in uncertain and complex scenes. Generally, precision and success rate are two metrics used to measure the robustness.

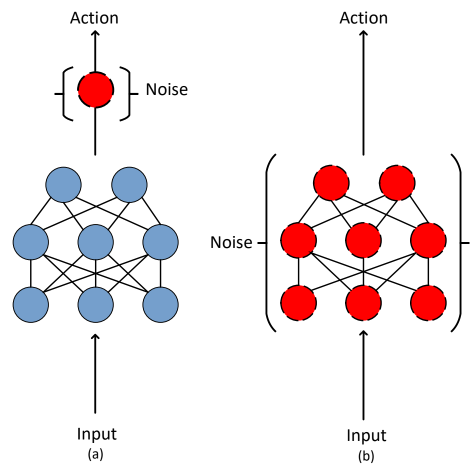

Fig. 1: Action space noise (a) and parameter space noise (b).

Existing vehicle tracking methods are mainly divided into correlation filter-based [6], [8], [9], deep learning-based [10], [11], and reinforcement learning (RL)-based [12], [13]. Correlation filter-based methods use correlation filter to learn the target features. During the tracking process, the candidate position with the maximum corresponding score is chosen as the prediction target position in the current frame. Owing to handcrafted features (e.g., Gray, Histogram of Oriented Gradient, and Color Names) and an efficient feature calculation in Fourier domain, correlation filter-based methods can perform real-time tracking. However, the representation ability of handcrafted features limits the tracking accuracy and robustness. With the rapid development of deep learning, deep features and deep network models are being introduced into tracking to improve the accuracy. Compared with the handcrafted features, deep features have better discriminative ability between the target and background. To accurately distinguish the target, complex deep models and massive feature calculations are demanded, which cannot meet the real-time requirements of vehicle tracking.

For RL-based methods, the tracking process is described as an evaluation function to provide the optimal action based on the current state. Different from correlation filter-based and deep learning-based methods which are static learning approaches [14], [15], RL is a trial-and-error process and belongs to dynamic learning, which seems more suitable for the vehicle tracking problem [16], [17]. However, existing RL-based trackers still have some issues. Compared with other

W. Xing, Y. Yang, S. Zhang, and Q. Yu are with the School of Software Engineering, Beijing Jiaotong University, Beijing 100044, China. Y. Yang is the corresponding author (e-mail: 16112088@bjtu.edu.cn).

L. Wang is with the School of Computer Science, University of Central Florida, Florida 32816, USA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

tracking targets, vehicles have specific characteristics such as fast speed and small deformation. Thus, they face the challenges of fast movement, occlusion, blur, and lighting changes in complex environments. The state-space dimension is much higher in vehicle tracking compared to the traditional reinforcement learning control tasks. For example, in the task of 'pendulum swing-up,' a vector with three coefficients is sufficient to represent the state space [16]. In vehicle tracking, the state vector's dimensionality may easily reach hundreds or thousands to allow the extracted deep features to represent video frames accurately [18]. Therefore, RL-based trackers are sensitive to random noise added to the action. For the vehicle tracking problem, the tracking model may lose the target under fast motion and occlusion due to random action noise. Since random action space noise directly affects the tracking results, the model may fluctuate dramatically and lose the vehicle targets, especially in complex environments.

As shown in Fig.1(a),The red point means random noise added to action and the blue points mean normal network parameters. The noise is random added to the output of network and most existing RL-based methods use action space noise to enhance the exploration ability of the model [19], [20]. However, the action space noise is random and cannot learn from current frame, which limits the robustness of the model. Moreover, most existing RL-based trackers use simple model update strategies or relocation algorithms, which will limit the robustness of the model in complex scenes, such as occlusion, blur, or deformation.

In this paper, a novel real-time robust vehicle tracking model called "NoisyOTNet" is proposed. Different from existing RL-based vehicle tracking models using action space noise, our proposed NoisyOTNet introduces noise into the parameter space to increase the exploration ability inspired by [19], [20], as shown in Fig.1 (b), different existing RL-based tracking method, the noise is added to the parameter of network.

Parameter space noise consists of two parts: a set of vectors of parameters, and a set of zero mean noise vectors generated by Gaussian distribution. In our method, the generated noise is added to the parameters of the fully connected layer, and the noisy parameters are updated during the tracking process. Furthermore, to enhance the robustness of the proposed NoisyOTNet model in complex scenes, an adaptive update strategy based on the UCB algorithm is proposed. It can adaptively update NoisyOTNet using the spatial-temporal information of the vehicle. Finally, an incremental learning-based relocation algorithm is designed to relocate a missing vehicle. The search area is adaptively scaled according to the size of the vehicle and background area to speed up the relocation process.

In this paper, we propose a novel real-time vehicle tracking model based on RL for traffic surveillance systems. The main contributions are as follows:

- A novel robust real-time vehicle tracking model, NoisyOTNet, is proposed, where parameter space noise is introduced. By formulating tracking as reinforcement learning with parameter space noise, the exploration ability of the model is improved.
- A new implementation for NoisyOTNet based on DDPGs with parameter noise is developed, which can better cope with the tracking task and directly predict the tracking result.
- An adaptive update strategy based on the UCB algorithm is proposed, which fully exploits the spatial-temporal information of the vehicle to adaptively update the tracking model.
- The proposed method is evaluated on the two popular tracking datasets UAV123 and OTB, and the results indicate that NoisyOTNet achieves good tracking performance.

The remainder of this paper is arranged as follows. We review the related studies in Section II. Section III describes the vehicle tracking problem based on RL, and Section IV expands the proposed vehicle tracking model, NoisyOTNet. Section V introduces the implementation of the model in training and online tracking. The experimental results are reported in Section VI. Finally, conclusions are drawn in Section VII.

II. Related Works

As a fundamental problem of traffic surveillance, numerous classic methods have been applied to vehicle tracking, including frame difference method, Gaussian mixture model, optical flow method, and correlation filter. With the great evolution of computing power, deep learning has accelerated the development of computer vision. In the following, we mainly review three types of deep learning methods: CNN-based methods, Siamese network-based methods, and deep reinforcement learning-based methods.

A. CNN-based Methods

CNN-based methods use deep features and deep networks to improve model performance. Deep features contain spatial information as well as rich semantic information. Compared with traditional feature representation, CNN-based methods achieve better representation and recognition ability. Fang et al. [21] designed a part-based AdaBoost tracking framework with weight relaxation factor to balance the sample weights. Hong et al. [22] introduced a deep convolutional network into tracking to improve the discriminative ability. Gao et al. [23] introduced an update-pacing framework with an ensemble of trackers to choose the most robust tracker for the remaining tracking. MDNet [24] is a multi-domain method to learn both common and specific domain features to represent a target, however, it suffers from an over-fitting problem owing to the huge size of the network. Song et al. [25] integrated Discriminative Correlation Filter (DCF) process into neural networks for end-to-end training, which combines correlation filters with CNNs to improve the tracking performance. Yuan et al. [26] defined the traffic force in tracking environment to describe the group behavior and handle complex interactions among vehicles. Bhat et al. [27] analyzed the complementary properties of deep and shallow features to improve the robustness of the model. As for the above CNN-based methods, some focus on achieving high tracking accuracy by designing complex network architectures [24], while the others pay attention to realizing real-time trackers by reducing the complexity of the model [25]. Compared with those previous works, we proposed an object tracking framework based on deep reinforcement learning, which can learn the pattern of object motion while tracking the target and improve tracking accuracy.

B. Siamese Network-based Methods

In recent years, Siamese networks [10], [11], [28], [29] have shown significant potential in tracking accuracy and speed, it accelerates the calculation process by sharing network weights. Tao et al. [15] introduced a Siamese network into tracking and compared the similarity between the search area and the template without model update to speed up the tracking process, which may lead to reduced tracking accuracy. Shan et al. [30] introduced multi-RPNs into the Siamese network and used FPN structure to build a detection subnetwork. Guo et al. [31] used an adaptive strategy to adapt to current target changes and increase the accuracy; however, the network structure in this method is shallow and has only a fewer layers, which may be not sufficient for tracking in the conditions of fast motion and occlusion. Zhu et al. [32] added a distractor-aware mechanism to improve the region proposal network-based tracker and increase its robustness and speed. However, the target variation feature in the tracking process was not utilized for model update, which reduced the accuracy of the model. Wang et al. [33] combined image segmentation with tracking to obtain a closer non-horizontal rectangular tracking bounding box to the real target, which further improves the accuracy of the model. Siamese networks prefer to speed up tracking by reducing update times. Therefore, an effective model update strategy is very important to identify a target and decrease the update overhead. To handle this issue, an adaptive model update strategy is designed based on the object changes and improves the model's discriminative ability.

C. Deep Reinforcement Learning-based Methods

Deep RL has been introduced in vehicle tracking with the development of deep learning in computer vision. In deep RL, the agent interacts with the environment and obtains the rewards constantly, and then the model is trained by maximizing the cumulative future rewards. Recently, there are some methods try to exploit the RL technique for vehicle tracking [12], [17], [34], [35]. Dong et al. [16] designed a continuous deep Q-learning model to track the target and used a regression method to solve the tracking problem. Yun et al. [12] applied a policy-based method to build the appearance model and classification model for target tracking. Huang et al. [36] analyzed the relationship between the network depth and prediction accuracy and designed a mechanism to adaptively adjust the depth of the computation to reduce the computational overhead. Supancic et al. [34] treated the target tracking process as a partially observable decision-making process and only updated the model when tracking drift occurs. This approach used an unlimited stream of Internet videos as the training samples. Liu et al. [37] utilized deep policy functions to determine the best action of the present state from a set of jump actions and learn the optimal policies. Chen et al. [13] introduced the Actor-Critic model and used networks for prediction and evaluation, however, only the action space noise and a simple relocation algorithm were used, which limited the robustness of the model. Ren et al. [17] used an iterative shift method and defined a new target evaluation mechanism to further distinguish the target and background. However, the robustness of the model was limited by the few training samples. For a non-real-time RL tracker, a better network architecture is necessary for high-speed robust tracking performance [12], [17], [38]. To tackle this issue, a parameter space noise is designed based on the current target, making the model jump out of locally optimal solutions and improving its robustness.

III. FRAMEWORK

In this paper, we propose a novel real-time vehicle tracking model, NoisyOTNet, based on deep RL.The proposed model introduces parameter space noise into RL for tracking to improve the robustness in complex scenes. NoisyOTNet consists of three main components: a parameter space noise-based tracking module, a spatial-temporal UCB based adaptive update module, and an adjustable incremental learning based relocation module.The proposed NoisyOTNet is based on deep deterministic policy gradients (DDPGs) framework, as shown in Fig.2. NoisyOTNet is implemented based on the Actor-Critic network, which consists of an Actor network and a Critic network. Based on the previous state of the target, the Actor predicts the optimal tracking result in the current frame. Then the Critic evaluates the obtained result. According to the evaluation result, the model is adaptively updated based on the UCB update strategy. Additionally, to address the vehicle missing problem, an effective relocation algorithm is designed to relocate the vehicle.

To introduce deep RL into the vehicle tracking problem, we define the vehicle tracking problem as MDP. NoisyOTNet is based on the Actor–Critic network by defining the vehicle tracking problem as a MDP. Our model includes agents, states, actions, state transitions, and rewards, as shown in Fig.2. The tracker selects an action a with a tracking reward $r(s,t)$ according to the current state s and uses the state transition function $s' = f(s,a)$ to continue the tracking. The state is represented by the input image M and the bounding box of the target (x, y, w, h). For an action, $a = (\Delta x, \Delta y, \Delta w, \Delta h)$ describes the vehicle movement. By applying action a to the original bounding box, we can obtain the new state $s' = (x', y', w', h')$ by

$$\begin{cases} x' = x + \Delta x * w \\ y' = y + \Delta y * h \\ w' = w + \Delta w * w \\ h' = h + \Delta h * h \end{cases} \tag{1}$$

For the reward, we use the Intersection-over-Union (IoU) criterion $IoU(GT, PB) = (GT \cap PB)/(GT \cup PB)$ between the ground truth (GT) and the predicted bounding box (PB) as the reward, which is commonly used in RL-based tracking methods [12].Hereby, we set the reward as $+1$ above a certain threshold and -1 below the threshold, respectively, aiming

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

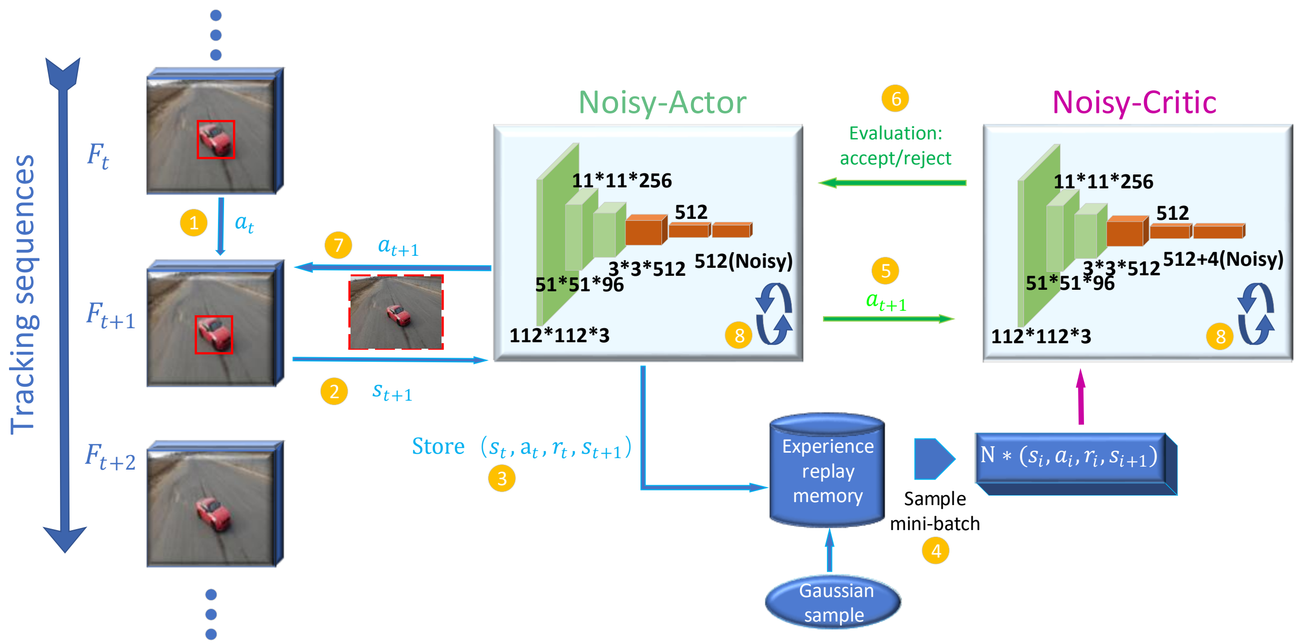

Fig. 2: The NoisyOTNet framework.

to keep the balance between positive and negative samples. The threshold is set as 0.7 which is an empirical value in the tracking field [12], [13]. The impact of different IoU thresholds are analyzed in Section VI. If the threshold is set smaller, more samples will be regarded as positive, which brings more label noise and difficulties in training convergence. If the threshold is set bigger, positive samples will become fewer, which breaks the balance of positive and negative samples and then degrades the accuracy of the model.

$$r(s,a) = \begin{cases} 1, & IoU(GT, PB) > 0.7 \\ -1, & IoU(GT, PB) \leqslant 0.7 \end{cases} \tag{2}$$

The pipeline of NoisyOTNet starts from choosing action and ends with updating of the model. The main steps are as follows:

Step 1: Initialize the target's state (i.e. the position and scale) in the current frame based on the tracking results in the previous frame.

Step 2: Pass the state and reward to the Noisy-Actor network.

Step 3: Use experience replay memory to store states, actions and rewards.

Step 4: Generate mini-batch samples by Gaussian sampling from the experience replay memory.

Step 5: Predict the action by Noisy-Actor and pass to the Noisy-critic.

Step 6: Evaluate the predicted action by Noisy-Critic to decide whether to accept the action. If the action is not accepted, the Noisy-Actor predicts a new action again until it is accepted.

Step 7: Predict the state of the target in the current frame by the obtained action.

Step 8: Update the model with the UCB update strategy.

IV. Method

In this section, we describe the parameter space noise, a parameter space noise based loss function, and an adaptive network update strategy and relocation algorithm.

A. Parameter Space Noise

Existing tracking models based on deep RL utilize action space noise to increase the exploration capability of a model in complex scenes. However, action space noise is randomly generated based on the current states, which are random and cannot be reproduced. Excessive randomness will lead to dramatic fluctuations of the model in complex scenes and the lost of the target. In addition, action space noise is approximately linear function and not easily incorporated with complicated functions, which lowers the performance of the model in complex scenes. To address this issue, parameter space noise is innovatively introduced into the proposed vehicle tracking model.Inserting the noise into the parameter space can enhance the exploration ability of the model [19], [20], which can further improve the robustness of tracking in complex scenes. This can improve the stability of the model in complex scenes while improving the exploration ability by generating richer behaviors.

Parameter space noise in a network disturbs its weights and deviations by noise parameter functions, and these noise parameters can be helpful for gradient descent. As for the input $\boldsymbol{x}$ and output $\boldsymbol{y}$, $\boldsymbol{y} = f_\theta(\boldsymbol{x})$ introduces interference by a vector of noisy parameters θ. The parameter space noise θ is defined as follows:

$$\theta = \mu + \Sigma \odot \epsilon, \tag{3}$$

where $\zeta = (\mu, \Sigma)$ indicates a set of vectors of parameters, ϵ is a vector of zero-mean noise with fixed statistics, and $\odot$ represents an element-wise multiplication. The loss of the network is represented by the expectation of noise, $\epsilon : \bar{L} = \mathbb{E}[L(\theta)]$. Thus, the loss function can be defined as an optimization of the set of parameters ζ.

The linear layer in the network can be expressed as follows:

$$\boldsymbol{y} = \omega^{\mathrm{T}} \boldsymbol{x} + b, \tag{4}$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the input, $\omega \in \mathbb{R}^{p \otimes q}$ is the weight matrix, and $b \in \mathbb{R}^q$ is the bias. Now, the linear layer is converted into a linear noise layer, which is defined as follows:

$$\boldsymbol{y} = (\mu^\omega + \sigma^\omega \odot \epsilon^\omega)^{\mathrm{T}} \boldsymbol{x} + \mu^b + \sigma^b \odot \epsilon^b, \tag{5}$$

where $\mu^\omega + \sigma^\omega \odot \epsilon^\omega$ and $\mu^b + \sigma^b \odot \epsilon^b$ replace ω and b, respectively. In addition, $\mu^\omega \in \mathbb{R}^{p \otimes q}, \mu^b \in \mathbb{R}^q, \sigma^\omega \in \mathbb{R}^{p \otimes q}$ and $\sigma^b \in \mathbb{R}^q$ are parameters, and $\epsilon^\omega \in \mathbb{R}^{p \otimes q}$ and $\epsilon^b \in \mathbb{R}^q$ are random noise variables.

There are two ways to generate noise [19], i.e. independent Gaussian noise and factorized Gaussian noise. The first is simpler to implement. Compared to the second, the first one is widely used in RL-based methods [19], [20]. In order to obtain better randomness of the noise parameters, we choose the independent Gaussian noise to generate noise for NoisyOTNet. The noise of each weight and bias are independent, and we use a unit Gaussian distribution to draw $\epsilon^\omega_{i,j}$ of the random matrix ϵ^ω (the same way as for ϵ^b_j of the random matrix ϵ^b).

By converting the linear layer into a linear noise layer, the network loss is changed into $\bar{L} = \mathbb{E}[L(\theta)]$, which is represented by the expectation of noise μ and ϵ. To obtain the gradients from a linear noise layer, the gradients are designed as follows:

$$\nabla \bar{L} = \nabla \mathbb{E}[L(\theta)] = \mathbb{E}[\nabla_{\mu,\Sigma} L(\mu + \Sigma \odot \epsilon)], \tag{6}$$

where we can obtain gradients from the linear noise layer based on the parameters μ and ϵ.

Furthermore, a Monte Carlo approximation is used for the noise gradients and the function takes ξ at each step of the optimization:

$$\nabla \bar{L} \approx \nabla_{\mu,\Sigma} L(\mu + \Sigma \odot \xi). \tag{7}$$

The above demonstrates how to convert the linear layer to a linear noise layer, and obtain the gradient information through the noise parameter. Then the parameter will be updated in the linear noise layer.

B. Noisy Deep Reinforcement Learning Model for Vehicle Tracking

In this section, we describe how to introduce a parameter space noise into the tracking model based on deep RL. We implement the proposed model based on the DDPG network structure. DDPG is an Actor–Critic network that can handle an action prediction in a continuous space. During the prediction, the Actor provides the results, whereas the Critic estimates the Q-value function using off-policy data and the recursive Bellman equation:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi_\theta(s_{t+1})), \tag{8}$$

where π_θ is the Actor. The Actor is trained to maximize the Q-values estimated by the Critic by back-propagating through both networks. For exploration, DDPG uses a stochastic policy of the form $\hat{\pi_\theta}(s_t) = \pi_\theta(s_t) + noise$, where an exploration is realized through an action space noise. In addition, the loss of the DDPG is defined as follows:

$$L(\theta) = \mathbb{E}\left[\mathbb{E}_{(\boldsymbol{x},a,r,\boldsymbol{y})\sim D}[Q(\boldsymbol{x}, a|\theta) - r - \gamma \underset{b\in A}{max} Q(\boldsymbol{y}, b|\theta^-)]^2\right], \tag{9}$$

where D is a distribution of transitions $e = (X, a, r = R(\boldsymbol{x}, a), \boldsymbol{y} \sim P(|\boldsymbol{x}, a))$ drawn from a replay buffer, and θ^- represents the parameters of a target network that updates $(\theta^- \leftarrow \theta)$ regularly to stabilize the learning.

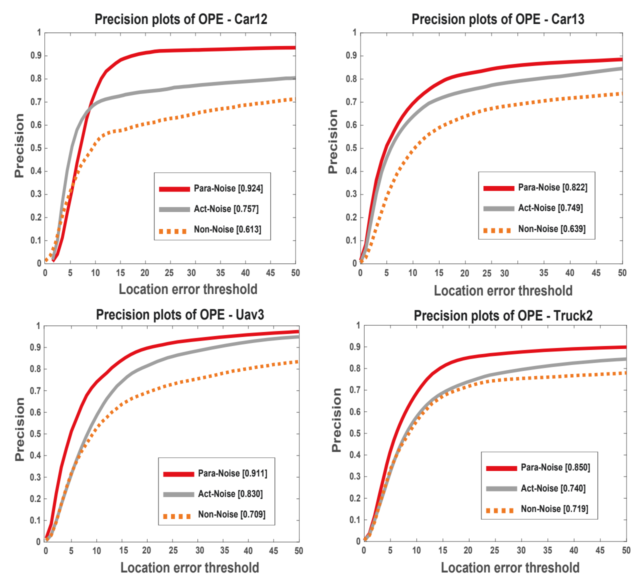

Fig. 3: Examples of tracking results of trackers with different noises.

As mentioned above, compared with action space noise, parameter space noise can improve the robustness of the model in complex scenes and can improve the exploration ability by generating richer behaviors.As shown in Fig 3, tracking models with noises can perform better than non-noise tracking models. Specifically, parameter space noise achieves more accurate tracking results than action space noise, indicating that the parameter space noise can improve the exploration ability of the model and the tracking robustness. Hence, we introduce a noise layer with the noise parameters into the DDPG network structure. By transforming the linear layer into a linear noise layer, we replace the action space noise with parameter space noise.The parameterized action-value function $Q(\boldsymbol{x}, a, \epsilon|\zeta)$ or $Q(\boldsymbol{x}, a, \epsilon'|\zeta^-)$ can be considered as a random variable, which is used to calculate the model loss:

$$\bar{L}(\zeta) = \mathbb{E}\left[\mathbb{E}_{(\boldsymbol{x},a,r,\boldsymbol{y})\sim D}[Q(\boldsymbol{x}, a, \epsilon|\zeta) - r - \gamma \underset{b\in A}{max} Q(\boldsymbol{y}, b, \epsilon'|\zeta^-)]^2\right], \tag{10}$$

where the external expectation is the distribution of the noise variable ϵ with respect to the noise value function

$Q(\boldsymbol{x}, a, \epsilon|\zeta)$ and noise variable ϵ' for a noisy target value function $Q(\boldsymbol{y}, b, \epsilon'|\zeta^-)$. Calculating the unbiased estimation of the loss is straightforward. For each transition in the replay buffer, we only need to calculate one instance of the target network and one instance of online network. We generate these independent noises to avoid deviations. Regarding the choice of action, another independent sample is generated for the online network [19].

C. Adaptive UCB-based Online Model Update Strategy

During the tracking process, the vehicle may change dramatically in complex scenes, such as deformation, occlusion, or blur. Thus, the model needs to be updated during the tracking process to deal with the changes of the current target.The traditional fixed update methods face difficulty in improving the validity of the model update in complex scenes and miss the target. Thus, an efficient model update strategy is critical to improve the robustness of the model in complex scenes.To address this problem, an adaptive model update strategy is designed based on the UCB, which uses online and target networks by considering the temporal information to adaptively update the model. For the proposed update strategy, we define four update policies as follows:

Non-update: The vehicle remains unchanged in the current scene, and the model can represent the vehicle well without an update, which can save time and improve the tracking speed.

Online update: A vehicle with minor changes can be represented by the model which needs to be updated to adapt to the vehicle in future frames. Only the parameters of the online network are used to update the model.

Online-target update: The vehicle undergoes dramatic changes in the current scene, and the model cannot represent the vehicle well. Thus, a single model update may miss the vehicle. Hence, an online model and a target model are used to update the model for adapting to the vehicle.

Relocation: The vehicle is out of view or occluded and cannot be tracked in the current searching area.We use an effective relocation algorithm to relocate the vehicle.

Compared with other update strategies based only on the model output, we innovatively introduce temporal information into the model update, which can improve the robustness of the model in complex scenes.

The UCB method can consider the temporal information by adding terms to the original update value Q_a and decide the model update strategy. The function is designed as follows:

$$UCB(v_a) = Q_a + c\sqrt{\frac{lnt}{N_t(a)}}, \tag{11}$$

where a indicates each update action, Q_a is the original value of a given by the model, c is a fixed weight parameter to balance the temporal information and the model's output, and $N_t(a)$ indicates the times this update action occurs in the previous t frames. The information is stored in a unified manner for model selection decisions.

The process of the adaptive UCB-based online model update strategy is shown in Algorithm 1.

Algorithm 1 Adaptive UCB-based online model update strategy.

Input: The original update value Q_a, UCB strategy interval t, UCB weight parameter c;
Output: Selected update strategy M;

Obtain the number of four model update actions $N_t(a_i)$ (i=1,2,3,4) from memory during t period;
  a_1 = Non-update;
  a_2 = Online update;
  a_3 = Online-target update;
  a_4 = Relocation;
$i = 1$;
repeat
  Calculate $c\sqrt{\frac{lnt}{N_t(a_i)}}$;
  $UCB(v_{a_i}) = Q_{a_i} + c\sqrt{\frac{lnt}{N_t(a_i)}}$;
  $i = i + 1$;
until Obtain all four model update action UCB values $UCB(v_{a_i})$ (i=1,2,3,4);
Select the model update action with the maximum $UCB(v_{a_i})$ as the current model update strategy M.
Update the model using the M model update strategy.
Update the memory with the model update strategy M.

By introducing the temporal information, the model chooses the update action based on both the current model's result and the temporal information generated by the model during the tracking process.Compared with the traditional model update method, the proposed update strategy increases the update actions and improves the robustness and real-time performance of the model in complex scenes.

D. Incremental Learning-based Relocation Algorithm

The lost of the vehicle tracking in complex scenes may be classified into two cases: 1) the vehicle is in the image, but out of the search area, and 2) the vehicle is out of the image. If the tracker cannot distinguish the two types of lost, it will cause the model to update with false samples, the target lost, or even sinking into an infinite loop.

To address this issue, we propose an effective incremental learning based relocation algorithm. It can effectively distinguish and relocate the above failure situations, and further improve the robustness of the model in a complex environment.

The proposed relocation algorithm can efficiently achieve local to global target relocation based on the target position, scale, and number of detection. Because the position and scale of the target are uncertain, the changes in the four dimensions, namely, top, down, left, and right, are different. First, we need to calculate the vertical variation Δh_{up} and Δh_{down}, as well as the horizontal variation Δw_{left} and Δw_{right} in the current frame based on the target position, scale, and number of detection D.

Taking the height changes as an example, $h_{up} = y_i - \frac{1}{2}h_i, h_{down} = H-(y_i - \frac{1}{2}h_i)$, the values of Δh_{up} and Δh_{down}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

Algorithm 2 Incremental learning-based relocation algorithm.

Input: The target position of the previous frame $P_{t-1}(x, y, w, h)$, the image size (W, H), the search time D, the current search time $d = 0$;
Output: Relocation location $P_t(x, y, w, h)$;

1: Obtain changes of scale in four directions: Δw_{left}, Δw_{right}, Δh_{up}, Δh_{down} ;
2: **repeat**
3: Expand the search area according to changes in scale;
4: Detect the target in the expanded search area and obtain a detection score;
5: **if** The detection score > 0.7 **then**
6: $P_t(x, y, w, h) = P_{relocation}(x, y, w, h)$;
7: Jump to 20;
8: **end if**
9: **if** The detection score > 0.3 **then**
10: $P_d(x, y, w, h)(d = 1, 2, ..., D) = P_{relocation}(x, y, w, h)$;
11: $d = d + 1$;
12: **else**
13: The current scale search fails, $d = d + 1$;
14: **end if**
15: **until** $d = D$;
16: **if** $P_t = \varnothing$ **then**
17: $P_t(x, y, w, h) = P_{t-1}(x, y, w, h)$;
18: **else**
19: Choose $P_d(x, y, w, h)$ with the maximum detection score as $P_t(x, y, w, h)$;
20: **end if**

can be calculated as follows:

$$\Delta h_{up} = \frac{h_{up}}{h_{up} + h_{down}} * \frac{1}{D},$$
$$\Delta h_{down} = \frac{h_{down}}{h_{up} + h_{down}} * \frac{1}{D}, \tag{12}$$

where Δh_{up} and Δh_{down} mean the update range for the top and bottom coordinates respectively. The relocation process will then use four directions as the update range to expand the search area until completing the relocation.

The incremental learning based relocation algorithm is shown in Algorithm 2.

Through the relocation algorithm, when the detection score is greater than 0.7 or the global-image detection has been completed, the tracking bounding box with the highest score is chosen for the result. If the highest score is still less than 0.3, the target is considered to be lost, and the previous frame result is assigned as the current frame prediction position. Then, the process continues the next frame tracking. The relocation can be performed efficiently when the target is lost. The search efficiency and speed are improved.

V. Implementation

We use ILSVRC (ImageNet Large Scale Visual Recognition Challenge) dataset [39] to pretrain NoisyOTNet for 250,000 iterations, which consists of 768 video sequences with bounding boxes. Specifically,NoisyOTNet has five convolutional layers and two fully connected layers, and the last fully connected layer is the noisy layer. Independent Gaussian noise is used for the noise generation. The noise of each weight and bias are independent, and both $\epsilon^{\omega}_{i,j}$ of the random matrix ϵ^{ω}, and ϵ^{b}_{j} of the random matrix ϵ^{b}, are drawn from a Gaussian distribution. For noise generation, we set μ and σ as follows.Each element $\mu_{i,j}$ is generated from independent uniform distributions $u[-\sqrt{\frac{3}{p}}, \sqrt{\frac{3}{p}}]$, where p is the number of linear layer inputs, and $\sigma_{i,j}$ is empirically set to 0.017 for all parameters [19], [40].

For online tracking, we set the update frequency for short and long term tracking of the Critic network to 10 and 100 respectively, using the last 10 and 30 frames to draw samples for the model update. In our experiment, 250 positive samples and 2,500 negative samples are generated around the ground truth in the first frame to initialize the tracking model. We generate 256 samples for the relocation process. The maximum number of steps for one frame is set to five, and the online model can adaptively decide how many steps for one frame should be applied based on the tracking results. The online tracking pipeline is shown in Algorithm 3.

Algorithm 3 NoisyOTNet pipeline for online tracking.

Input: Initial target position P_0 and image;
Output: Estimated target position $P_t = (x_t, y_t, w_t, h_t)$;

1: Generate samples in the first frame to update a noisy network;
2: **repeat**
3: Extract features from (x_{t-1}, y_{t-1});
4: **repeat**
5: Apply the Actor network to give the predicted position using single or multi-step tracking;
6: Apply the Critic network with the position and features to obtain the score;
7: **if** relocation **then**
8: Use the relocation model to find a position with a higher score around the bounding box;
9: **end if**
10: **until** End of the current frame tracking;
11: Update the model using the predicted position $P_t = (x_t, y_t, w_t, h_t)$ with the adaptive update strategy;
12: **until** End of video sequence.

VI. Experiments

The proposed vehicle tracking model is implemented by using the Pytorch toolkit. We use a computer with 3.4 GHz 7700k CPU, 11 GB GTX1080Ti graphics card, and 32 GB of memory to train and test the vehicle tracking model. The proposed model can achieve a frame rate of 41 FPS during online tracking on average, which meets the requirement of real-time vehicle tracking.

We adopt both precision and success plots to evaluate the performance of the trackers. Precision is defined based on the distance between the predicted location and the ground truth as: $dist = \sqrt{(G_x - P_x)^2 + (G_y - P_y)^2}$. If the distance is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

TABLE I: Video challenging attributes on UAV123. Each video may have more than one challenging attribute.

Video challenging attribute	Description
Aspect ratio change (ARC)	The fraction of the ground truth aspect ratio in the first frame and at least one subsequent frame is outside the range [0.5, 2].
Background clutter (BC)	The background near the target has a similar appearance as the target.
Camera motion (CM)	Abrupt motion of the camera.
Fast motion (FM)	The motion of the ground truth bounding box is larger than 20 pixels between two consecutive frames.
Full occlusion (FOC)	The target is fully occluded.
Illumination variation (IV)	The illumination of the target changes significantly.
Low resolution (LR)	At least one ground truth bounding box has less than 400 pixels.
Out of view (OV)	Some portion of the target leaves the view.
Partial occlusion (POC)	The target is partially occluded.
Similar target (SOB)	There are targets of a similar shape or same type near the target.
Scale variation (SV)	The ratio of the initial and at least one subsequent bounding box.
Viewpoint change (VC)	Viewpoint affects target appearance significantly.

TABLE II: Evaluation results of trackers on UAV123. The proposed NoisyOTNet achieves comparable results with state-of-the-art trackers. The best results are given in bold.

Tracker	NoisyOTNet	ECO	ARCF	AutoTrack	ADNet	ACT	MIMRT	TSD	KAOT
Precision	**0.762**	0.741	0.666	0.671	0.720	0.692	0.726	0.659	0.686
AUC	**0.525**	**0.525**	0.506	0.473	0.510	0.496	0.484	0.464	0.479
FPS	41	8	15	**60**	8	35	5	42	15
Real-time	Y	N	N	Y	N	Y	N	Y	N
Deep Learning	Y	Y	Y	Y	Y	Y	Y	Y	Y
Programming Language	Python	Matlab	Python	Python	Matlab	Python	Python	Python	Python

smaller than a predefined threshold, the tracking in the frame is considered to be precise. Thus precision is defined as the percentage of the number of frames in which the distance is smaller than the threshold and the total frame number. Success rate is calculated by the IoU score. The IoU can be defined as: $IOU(GT, PB) = (GT \cap PB)/(GT \cup PB)$. If the IoU value is larger than a predefined threshold in one frame, the tracking in that frame is taken as successful. Success rate is defined as the percentage of the number of successful frames and the total frame number. Commonly, the precision threshold is set as 20 pixels and the success rate threshold is set as 0.5. In addition, the Area Under the Curve (AUC) of the success plot is also used as another metric.

A. Evaluation on UAV123

UAV123 [41] is an unmanned aerial vehicle (UAV) tracking dataset as a widely used benchmark in the field of vehicle tracking. It contains 123 fully annotated HD video sequence data captured from a low-altitude aerial perspective, including 115 videos clips captured by a drone camera and 8 video sequences rendered by a UAV simulator. To refine the tracking scene, all video sequences are refined according to 12 common challenging attributes, as shown in Table I. The trackers are evaluated in terms of tracking accuracy and run-time.

As shown in Table II,we conducted comparison with eight state-of-the-art trackers (ECO [7], ARCF [42], AutoTrack [43], ADNet [12], ACT [13], MIMRT [44], TSD [45], KAOT [46]). NoisyOTNet achieves better performance in terms of both the precision and AUC plots. The precision rate of NoisyOTNet is 76.2% and 2.1% higher than the performance of the second-best tracker ECO. The success rate of NoisyOTNet and ECO are both 52.5%, outperforming the other trackers, and Noisy-OTNet tracks at 41 FPS, while ECO runs at 8 FPS. UAV123 contains many small targets which are hard to track in low-resolution videos. NoisyOTNet performs well on small targets, as shown in Section VI-E. The experiment results demonstrate that the proposed method can achieve efficient and real-time tracking in complex scenes.

TABLE III: Ablation study of different components on UAV123. The best results are shown in bold.

Noisy	UCB	IR	Precision	Success
			71.1%	47.6%
√			74.6%	51.2%
	√		73.4%	50.1%
		√	72.8%	49.3%
√	√		75.7%	51.9%
√		√	75.1%	51.7%
	√	√	74.2%	50.8%
√	√	√	**76.2%**	**52.5%**

B. Ablation Study of Different Components

To demonstrate the impact of the components in Noisy-OTNet, we applied three variants of our tracker by integrating a network with different types of update and relocation strategies and evaluated them on the UAV123 dataset. The baseline model is a model without “Noisy,” “UCB,” or “IR” components. These three variants are as follows: 1) “Noisy” is a baseline model that contains a parameter space noise network; 2) “UCB” is the baseline model guided by the adaptive UCB-based online update strategy, which contains four policies, namely, non-update, online update, online-target update, and relocation; and 3) “IR” is the baseline model with a relocation algorithm based on incremental learning.We test the three components separately to verify their effectiveness respectively. Table III shows the precision and success plots of these variations on the UAV123 dataset.

1) Parameter Space Noise: From Table III, we can see that compared with the base model the “Noisy” component significantly improves the precision and success rate by 3.5% and 3.6%, respectively. The reason for this improvement is the fact that the baseline model uses random action space noise to enhance the model exploration capabilities. When the target changes drastically in complex scenes, the model cannot smoothly adapt to the change of the target. Compared with action space noise, parameter space noise consists of two parts, a parameter μ and noise ϵ, and can update based on the current tracking result. While enhancing the exploration capability of the model, it can also maintain a stable update and adapt to the target in complex scenes, such as background clutter, scale variation, and partial occlusion.

2) Adaptive UCB-based online model update strategy: Compared with the baseline model, the UCB variants are 2.3% and 2.5% higher in terms of the precision and success rate, respectively. The baseline model uses a fixed update strategy, which cannot be adaptively adjusted according to target changes. The UCB uses an adaptive update based on the scene and target changes to improve the efficiency. We also designed four model update policies: non-update, online update, online-target update, and relocation. Different update policies are selected according to the UCB result, which considers both the current tracking result and spatial-temporal information during tracking. The results indicate that it can improve the efficiency of the model update as well as the accuracy of the tracking in complex scenes.

3) Incremental learning-based relocation algorithm: The baseline model does not have relocation algorithm, and when partial occlusion, complete occlusion, or out of view occurs, it may lose the target, and result in tracking failure. We conducted local-to-global relocation instead of a simple global-image relocation, which considers the target motion, size, and background size. Through an effective relocation, the model can judge the case of the current lost target and choose the optimal relocation result as the current target tracking position. As shown in Table III, the “IR” component can effectively improve both the precision and success rates by 1.7% based on the baseline model.

4) NoisyOTNet: “Noisy+UCB” enables the adaptive UCB-based online update strategy to update the model based on the environment and target changes, which can effectively reduce the risk of drift. It outperforms “Noisy” by 1.1% on the precision plot and 0.7% on the success plot. Moreover, “Noisy+UCB+IR” combines all optimizations of NoisyOTNet,

TABLE IV: Impact of different IoU thresholds on tracking performance of the proposed method on the OTB-2015 dataset. The best results are highlighted in bold.

IoU	0.5	0.6	0.7	0.8	0.9
Precision	0.811	0.837	**0.902**	0.871	0.791
AUC	0.575	0.603	**0.672**	0.636	0.553

based on “Noisy+UCB”. NoisyOTNet adopts an incremental learning based relocation algorithm instead of the simple relocation algorithm. Our strategy can relocate the missing target more efficiently and achieve 1.2% and 0.6% performance gains in terms of the precision over the ”Noisy” and ”Noisy+UCB” on a precision plot, respectively.

5) IoU Thresholds: In order to further analyze the impact of different parameters of IoU on tracking performance, we set different IoU thresholds. We validated them on the OTB-2015 dataset, and the experimental results are shown in the Table IV. If the threshold is too small, some of the negative samples may be turned into positive, which cannot effectively distinguish the target from the background when the background is similar to the target, resulting in tracking failure. If the threshold is too large, the number of positive samples will be reduced, enlarging the imbalance between positive and negative samples, which may reduce the model’s accuracy. Therefore, we use the threshold of 0.7, which allows the proposed method to achieve better tracking performance.

C. Evaluation on OTB

In order to show the robustness of the proposed method, we conduct the experiment on OTB dataset by using the same hyperparameter settings as the UAV123 dataset. OTB [47], [48] includes 100 video sequences and is widely used in the object tracking field. It is subdivided into OTB-2013 and OTB-2015 with 51 and 100 tracking sequences, respectively. Same with the UAV123 experiments, precision and AUC rates are used to evaluate the performance of trackers. We use precision-13 and AUC-13 for OTB-2013 dataset, and precision-15 and AUC-15 for OTB-2015 dataset to evaluate the accuracy. The comparison results with 11 state-of-the-art trackers (ADNet [12], ACT [13], HP [16], DRL-IS [17], EAST [36], UCT [32], CREST [25], SiamDW [28], SiamRPN++ [10], C-RPN [11], and RASNet [33]) are shown in Table V.

From Table V we can observe that NoisyOTNet performs better than reinforcement learning-based trackers including ADNet, HP, DRL-IS, EAST, and ACT on both benchmark datasets. ADNet and ACT use action space noise to increase the exploration ability of the model. HP uses deep Q-Learning to enhance the exploration, and DRL-IS designs an update module to increase the exploration ability of the model. Compared with ADNet that achieves 89.6% and 88.0% on precision-13 and precision-15, NoisyOTNet performs 2.9% and 2.2% higher than ADNet, and 3 times faster than ADNet. ACT is a real-time RL-based tracker, NoisyOTNet performs 4.1% and 4.3% higher than ACT on precision-13 and precision-15, respectively. The results show that the proposed method performs competitively against the state-of-the-art RL-based methods.

TABLE V: Evaluation results of trackers on OTB. The best and second best results are denoted in bold and underline, respectively.

	Ours	UCT	CREST	SiamDW	SiamRPN++	C-RPN	RASNet	HP	DRL-IS	ADNet	ACT	EAST
Precision-13	**0.925**	0.904	0.908	0.88	0.918	0.897	0.892	0.841	0.923	0.896	0.884	0.851
AUC-13	**0.685**	0.641	0.673	0.666	0.68	0.675	0.67	0.629	0.682	0.672	0.667	0.638
Precision-15	0.902	0.849	0.837	0.854	**0.903**	0.871	0.857	0.796	0.901	0.88	0.859	0.813
AUC-15	**0.672**	0.611	0.623	0.64	0.67	0.663	0.642	0.601	0.671	0.668	0.648	0.612
Real-Time	45	41	10	35	35	32	83	6.9	10.2	8	35	**159**

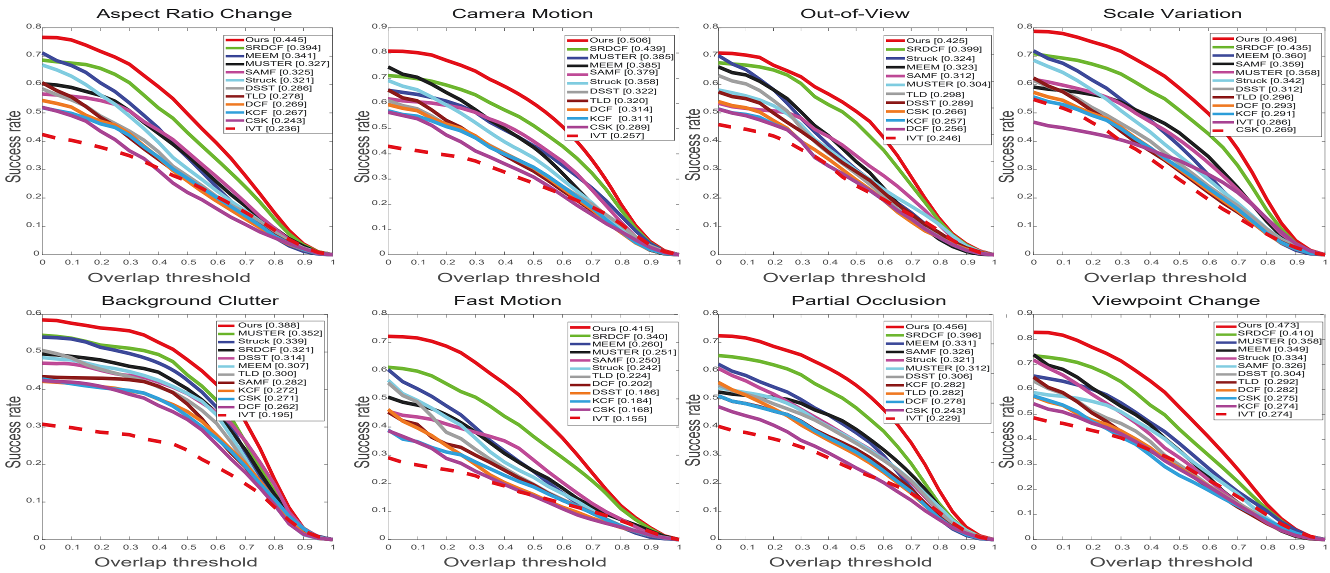

Fig. 4: AUC scores of different attributes: aspect ratio change (ARC), camera motion (CM), out of view (OV), scale variation (SV), background clutter (BC), fast motion (FM), partial occlusion (POC), and viewpoint change (VC).

TABLE VI: The real-time performance of the proposed method on the UAV123 and OTB-2015 datasets.

Dataset	UAV123	OTB-2015
Average Tracking Speed (Frame Per Second, FPS)	41	45

In order to further analyze the real-time performance of the proposed method, seven real-time trackers mentioned in Section II are employed for comparison on OTB dataset, and the results are shown in Table V. It can be found that the proposed method achieves the real-time performance with speed 45 FPS, which is faster than most competing trackers and only some slower than RASNet and EAST. Thus it shows that our method achieves higher accuracy while the model maintains competitive real-time performance compared to the state-of-the-art trackers.

In addition, we also discuss the influence of different exploration strategies. In Table V, HP [16] is a Q-learning-based approach to learn hyperparameters to improve the exploration capability of the model. ADNet [12] is a policy-based method that uses stochastic strategy to enhance the exploration capability. EAST is implemented based on DQN to achieve efficient exploration of the model. ACT [13] expands the exploration capability by adding action noise. DRL-IS [17] follows the Actor-Citric framework and designs different update strategies to expand the search space of the model. The results show that the proposed method with parameter noise can outperforms the RL-based trackers with other exploration strategies on these two datasets.

To analyze the proposed method's real-time performance, we have added the tracking speed experiments on the UAV123 and OTB-2015 datasets, as shown in Table VI. The results show that the proposed method achieves the 41 FPS and 45 FPS, respectively in these two datasets and can meet the real-time requirement.

D. Quantitative Analysis

We conducted a comparison with 11 state-of-the-art trackers (SRDCF, SAMF, MUSTER [50], DSST, Struck [51], DCF [49], KCF [6], CSK [52], TLD [53]), and the results are shown in Fig. 4.

Fig. 4 shows the AUC of the different trackers for eight challenging attributes in UAV123. The results show that the NoisyOTNet method performs well on all of these challenging attributes.

For camera motion and viewpoint changes, we achieved the highest score among the state-of-the-art trackers, 6.7% and 6.3% higher than the second-best tracker SRDCF, respectively. Because the parameter space noise introduced is reproducible, it can improve the model's ability to explore while enhancing the robustness of the model in complex scenes. The experimental results of the aspect ratio changes and background clutter demonstrate that our proposed model

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

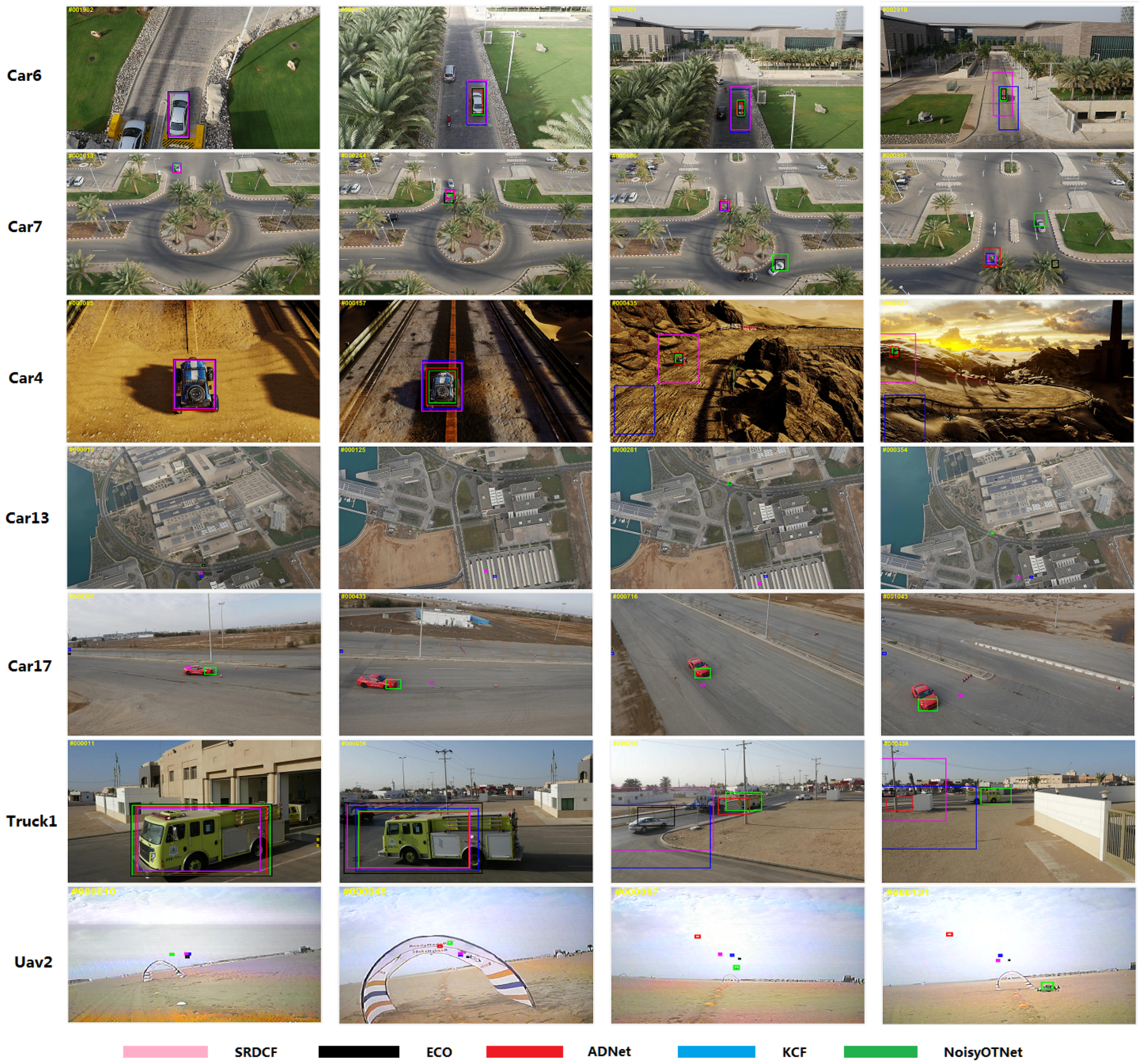

Fig. 5: Qualitative evaluation of the proposed NoisyOTNet, SRDCF, KCF, ADNet, and ECO on seven challenging sequences, *Car6*, *Car7*, *Car4*, *Car13*, *Car17*, *Truck1*, and *Uav2*.

performs better than the other state-of-the-art trackers. For the RL-based trackers, the model can apply the original knowledge into a new environment and adapt the model to the current tracking target.

Furthermore, SRDCF and Struck use a fixed update strategy, and when the target changes drastically under occlusions and fast movements, the updated model cannot represent the target well. Compared with these approaches, the adaptive update strategy can improve the robustness of the updated model. The updated model can track the target after such drastic changes, and achieves better performance on partial occlusions and fast motions. For out-of-view attribute, NoisyOTNet benefits from the relocation algorithm and achieves a rate of 42.5% on the success plot, and can judge whether the target is lost when it moves out of image quickly without significant calculation overhead.

The experimental results demonstrate that the proposed vehicle tracker can accurately track the vehicle target in complex scenes under occlusions, out of view, and deformations, and has better robustness than the other state-of-the-art trackers used in the experiment.

E. Qualitative Evaluation

Fig. 5 shows the tracking results of several top tracking methods including MDNet, KCF, ADNet, CF2, and our proposed method on seven challenging sequences. These challenges include scale changes, occlusions, viewpoint changes, small targets, deformations, and low resolution. We evaluate the robustness of our proposed model based on the experimental results on these video sequences.

In sequence Car6, the scale of the vehicle has changed significantly, and ECO and NoisyOTNet can cope with the scale change of the current vehicle well. ADNet is sensitive to the changes of the target because the action space noise is based on the predictions. When the target scale changes significantlyit will be less robustness to scale changes. In sequence Car7, the vehicle completely occludes twice in complex scenes, and KCF, ADNet, and SRDCF lose the target after the first occlusion; ECO loses the target after the second occlusion; and NoisyOTNet can still maintain the correct positioning of the target after two occlusions because when the target is occluded or lost, the relocation algorithm allows our tracker to quickly and accurately relocate the target when it reappears.

In sequence Car4, the target scale and background change drastically, and KCF and SRDCF completely lose the target, whereas the other trackers can still track the target.They use deep features with stronger discriminative ability for representation, improving the tracking robustness in complex scenes. In the sequences of Truck1 and Car17, the vehicle is deformed and flipped. In the case of deformation, the other trackers lose the target during tracking based on a fixed and single update strategy. NoisyOTNet can effectively track the target in complex scenes by updating the model with the adaptive update strategy. The targets in Uav7 and Car13 are small with low resolution. These small targets also suffer from an uncertain motion trajectory and full occlusions. As shown in Fig. 5, SRDCF and ADNet gradually lose their targets during the tracking process, whereas ECO loses the target when it becomes extremely small. The experimental results indicate that the proposed tracker can track small targets in low-resolution scenes.

Through the above qualitative analysis, we can see that in complex scenes, such as scale changes, occlusions, complex background, small targets, and low resolution, our proposed model can achieve robust tracking. Meanwhile, compared with ECO and ADNet, which perform at 8 FPS, our model can run at 41 FPS on UAV123 and meet the real-time vehicle tracking requirement.

VII. Conclusions

In this paper, we propose a novel real-time vehicle tracking model NoisyOTNet, which enables accurate vehicle tracking in complex scenes. The parameter space noise introduced into the proposed model is different from action space noise used by existing RL models. The parameter space noise consists of parameters and noise and can improve the robustness and exploration capabilities of the model in complex scenes. Furthermore, the adaptive online update strategy learns the spatial-temporal information and select the optimal update policy to quickly and accurately update the model. The updated model can accurately represent the target after dramatic changes in complex scenes. A relocation algorithm based on incremental learning is also proposed to relocate lost target in complex scenes. Finally, the experimental results on UAV123 and OTB datasets verify that NoisyOTNet can effectively conduct real-time tracking in complex scenes and achieve competitive results compared with other state-of-the-art RL methods. The current model can also be further optimized. As future work, we will introduce an adaptive relocation method, and upgrade the current network structure with better deep learning model to further improve the robustness and speed of the proposed model.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61876018, No.61976017 and No.61601021), the Beijing Natural Science Foundation (No.4202056, No.4212025 and L172022).

References

[1] Tian, Bin and Morris, Brendan Tran and Ming, Tang and Liu, Yuqiang and Yao, Yanjie and Chao, Gou and Shen, Dayong and Tang, Shaohu, Hierarchical and Networked Vehicle Surveillance in ITS: A Survey, IEEE Trans. Intell. Transp. Syst., 2017, 18(1): 25-48.

[2] Jhonghyun An and Baehoon Choi and Hyunju Kim and Euntai Kim, A New Contour-Based Approach to Moving Object Detection and Tracking Using a Low-End Three-Dimensional Laser Scanner, IEEE Trans. Veh. Technol., 2019, 68(8): 7392-7405.

[3] Wang, Qiang and Zhang, Li and Bertinetto, Luca and Hu, Weiming and Torr, Philip HS, Fast Online Object Tracking and Segmentation: A Unifying Approach, In Proc. IEEE Int. Conf. Comput. Vis., 2018: 1328-1338.

[4] Jie, Wang and Xiao, Zhang and Gao, Qinghua and Hao, Yue and Wang, Hongyu, Device-free Wireless Localization and Activity Recognition: A Deep Learning Approach, IEEE Trans. Veh. Technol., 2017, 66(7): 6258-6267.

[5] Driusso, Marco and Marshall, Chris and Sabathy, Mischa and Knutti, Fabian and Mathis, Heinz and Babich, Fulvio, Vehicular Position Tracking Using LTE Signals, IEEE Trans. Veh. Technol., 2017, 66(4): 3376-3391.

[6] Henriques, João F and Caseiro, Rui and Martins, Pedro and Batista, Jorge, High-Speed Tracking With Kernelized Correlation Filters, IEEE Trans. Pattern Anal. Mach. Intell., 2015, 37(3): 583-596.

[7] Danelljan, Martin and Bhat, Goutam and Khan, Fahad Shahbaz and Felsberg, Michael, ECO: Efficient Convolution Operators for Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 6931–6939.

[8] Valmadre, Jack and Bertinetto, Luca and Henriques, João and Vedaldi, Andrea and Torr, Philip HS, End-to-End Representation Learning for Correlation Filter Based Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 2805-2813.

[9] Tianzhu, Zhang and Changsheng, Xu and Ming-Hsuan, Yang, Learning Multi-Task Correlation Particle Filters for Visual Tracking, IEEE Trans. Pattern Anal. Mach. Intell., 2019, 41(2): 365-378.

[10] Li, Bo and Wu, Wei and Wang, Qiang and Zhang, Fangyi and Xing, Junliang and Yan, Junjie, SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks, In Proc. Eur. Conf. Comput. Vis., 2019: 4282-4291.

[11] Fan, Heng and Ling, Haibin, Siamese Cascaded Region Proposal Networks for Real-Time Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2019: 7952-7961.

[12] Yun, Sangdoo and Choi, Jongwon and Yoo, Youngjoon and Yun, Kimin and Choi, Jin Young, Action-Decision Networks for Visual Tracking With Deep Reinforcement Learning, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 1349-1358.

[13] Chen, Boyu and Wang, Dong and Li, Peixia and Wang, Shuang and Lu, Huchuan, Real-Time 'Actor-Critic' Tracking, In Proc. Eur. Conf. Comput. Vis., 2018: 328-345.

[14] David S. Bolme and J. Ross Beveridge and Bruce A. Draper and Yui Man Lu, Visual Object Tracking Using Adaptive Correlation Filters, In Proc. IEEE Conf. Comput. Vis. and Pattern Recogn., 2010: 2544-2550.

[15] Tao, Ran and Gavves, Efstratios and Smeulders, Arnold WM, Siamese Instance Search for Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2016: 1420-1429.

[16] Dong, Xingping and Shen, Jianbing and Wang, Wenguan and Liu, Yu and Shao, Ling and Porikli, Fatih, Hyperparameter Optimization for Tracking With Continuous Deep Q-Learning, In Proc. IEEE Int. Conf. Comput. Vis., 2018: 518-527.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2021.3086104, IEEE Transactions on Circuits and Systems for Video Technology

[17] Ren, Liangliang and Yuan, Xin and Lu, Jiwen and Yang, Ming and Zhou, Jie, Deep Reinforcement Learning With Iterative Shift for Visual Tracking, In Proc. Eur. Conf. Comput. Vis., 2018: 684-700.

[18] David Silver and Guy Lever and Nicolas Heess and Thomas Degris and Daan Wierstra and Martin A. Riedmiller, Deterministic Policy Gradient Algorithms, In Proc. Int. Conf. on Mach. Learn., 2014, 32: 387-395.

[19] Meire Fortunato and Mohammad Gheshlaghi Azar and Bilal Piot and Jacob Menick and Matteo Hessel and Ian Osband and Alex Graves and Volodymyr Mnih and Rémi Munos and Demis Hassabis and Olivier Pietquin and Charles Blundell and Shane Legg, Noisy Networks For Exploration, In Proc. Int. Conf. Learn. Represent., 2018.

[20] Matthias Plappert and Rein Houthooft and Prafulla Dhariwal and Szymon Sidor and Richard Y. Chen and Xi Chen and Tamim Asfour and Pieter Abbeel and Marcin Andrychowicz, Parameter Space Noise for Exploration, In Proc. Int. Conf. Learn. Represent., 2018.

[21] Jianwu Fang and Qi Wang and Yuan Yuan, Part-Based Online Tracking With Geometry Constraint and Attention Selection, IEEE Trans. Circuits Syst. Video Technol., 2014, 24(5): 854-864.

[22] Hong, Seunghoon and You, Tackgeun and Kwak, Suha and Han, Bohyung, Online Tracking by Learning Discriminative Saliency Map With Convolutional Neural Network, In Proc. IEEE Int. Conf. Mach. Learn., 2015: 597-606.

[23] Yuefang Gao and Zexi Hu and Henry Wing Fung Yeung and Yuk Ying Chung and Xuhong Tian and Liang Lin, Unifying Temporal Context and Multi-Feature With Update-Pacing Framework for Visual Tracking, IEEE Trans. Circuits Syst. Video Technol., 2020, 30(4): 1078-1091.

[24] Nam, Hyeonseob and Han, Bohyung, Learning Multi-Domain Convolutional Neural Networks for Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2016: 4293-4302.

[25] Song, Yibing and Ma, Chao and Gong, Lijun and Zhang, Jiawei and Lau, Rynson WH and Yang, Ming-Hsuan, Crest: Convolutional Residual Learning for Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 2574-2583.

[26] Yuan Yuan and Yuwei Lu and Qi Wang, Tracking as a Whole: Multi-Target Tracking by Modeling Group Behavior With Sequential Detection, IEEE Trans. Intell. Transp. Syst., 2017, 18(12): 3339-3349.

[27] Bhat, Goutam and Johnander, Joakim and Danelljan, Martin and Shahbaz Khan, Fahad and Felsberg, Michael, Unveiling the power of deep tracking, In Proc. Eur. Conf. Comput. Vis., 2018: 483-498.

[28] Zhipeng, Zhang and Houwen, Peng and Qiang, Wang, Deeper and Wider Siamese Networks for Real-Time Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2019: 4591-4600.

[29] Dongdong Li and Fatih Porikli and Gongjian Wen and Yangliu Kuai, When Correlation Filters Meet Siamese Networks for Real-Time Complementary Tracking, IEEE Trans. Circuits Syst. Video Technol., 2020, 30(2): 509-519.

[30] Yunxiao Shan and Xiaomei Zhou and Shanghua Liu and Yunfei Zhang and Kai Huang, SiamFPN: A Deep Learning Method for Accurate and Real-Time Maritime Ship Tracking, IEEE Trans. Circuits Syst. Video Technol., 2021, 31(1): 315-325.

[31] Guo, Qing and Feng, Wei and Zhou, Ce and Huang, Rui and Wan, Liang and Wang, Song, Learning Dynamic Siamese Network for Visual Object Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 1781-1789.

[32] Zhu, Zheng and Huang, Guan and Zou, Wei and Du, Dalong and Huang, Chang, UCT: Learning Unified Convolutional Networks for Real-time Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis. Workshops, 2017: 1973-1982.

[33] Wang, Qiang and Teng, Zhu and Xing, Junliang and Gao, Jin and Hu, Weiming and Maybank, Stephen, Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2018: 4854-4863.

[34] Supancic III, James Steven and Ramanan, Deva, Tracking as Online Decision-Making: Learning a Policy from Streaming Videos With Reinforcement Learning, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 322-331.

[35] Zhenjun Han and Pan Wang and Qixiang Ye, Adaptive Discriminative Deep Correlation Filter for Visual Object Tracking, IEEE Trans. Circuits Syst. Video Technol., 2020, 30(1): 155-166.

[36] Huang, Chen and Lucey, Simon and Ramanan, Deva, Learning Policies for Adaptive Tracking With Deep Feature Cascades, In Proc. IEEE Int. Conf. Comput. Vis., 2017: 105-114.

[37] Xiaobai Liu and Qian Xu and Thuan Chau and Yadong Mu and Lei Zhu and Shuicheng Yan, Revisiting Jump-Diffusion Process for Visual Tracking: A Reinforcement Learning Approach, IEEE Trans. Circuits Syst. Video Technol., 2019, 29(8): 2431-2441.

[38] Chen, Boyu and Wang, Dong and Li, Peixia and Wang, Shuang and Lu, Huchuan, Real-Time 'Actor-Critic'Tracking, In Proc. Eur. Conf. Comput. Vis., 2018: 328-345.

[39] Russakovsky, Olga and Deng, Jia and Su, Hao and Krause, Jonathan and Satheesh, Sanjeev and Ma, Sean and Huang, Zhiheng and Karpathy, Andrej and Khosla, Aditya and Bernstein, Michael and others, Imagenet Large Scale Visual Recognition Challenge, Int. Jour. Comput. Vis., 2015, 115(3): 211-252.

[40] Fortunato, Meire and Blundell, Charles and Vinyals, Oriol, Bayesian Recurrent Neural Networks, CoRR, 2017: abs/1704.02798.

[41] Mueller, Matthias and Smith, Neil and Ghanem, Bernard, A Benchmark and Simulator for Uav Tracking, In Proc. Eur. Conf. Comput. Vis., 2016: 445-461.

[42] Ziyuan Huang and Changhong Fu and Yiming Li and Fuling Lin and Peng Lu, Learning Aberrance Repressed Correlation Filters for Real-Time UAV Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2019: 2891-2900.

[43] Yiming Li and Changhong Fu and Fangqiang Ding and Ziyuan Huang and Geng Lu, AutoTrack: Towards High-Performance Visual Tracking for UAV With Automatic Spatio-Temporal Regularization, In Proc. IEEE Conf. Comput. Vis. and Pattern Recogn., 2020: 11920-11929.

[44] Yufei Zha and Yuanqiang Zhang and Tao Ku and Hanqiao Huang and Wei Huang and Peng Zhang, Multiple Instance Models Regression for Robust Visual Tracking, IEEE Trans. Circuits Syst. Video Technol., 2021, 31(3): 1125-1137.

[45] Fan Li and Changhong Fu and Fuling Lin and Yiming Li and Peng Lu, Training-Set Distillation for Real-Time UAV Object Tracking, In Proc. IEEE Int. Conf. on Robot. Autom., 2020: 9715-9721.

[46] Yiming Li and Changhong Fu and Ziyuan Huang and Yinqiang Zhang and Jia Pan, Keyfilter-Aware Real-Time UAV Object Tracking, In Proc. IEEE Int. Conf. on Robot. Autom., 2020: 193-199.

[47] Wu, Yi and Lim, Jongwoo and Yang, Ming-Hsuan, Online Object Tracking: A Benchmark, In Proc. IEEE Int. Conf. Comput. Vis., 2013: 2411-2418.

[48] Wu, Yi and Lim, Jongwoo and Yang, Ming Hsuan, Object Tracking Benchmark, IEEE Trans. Pattern Anal. Mach. Intell., 2015, 37(9): 1834-1848.

[49] Danelljan, Martin and Hager, Gustav and Shahbaz Khan, Fahad and Felsberg, Michael, Learning Spatially Regularized Correlation Filters for Visual Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2015: 4310-4318.

[50] Hong, Zhibin and Chen, Zhe and Wang, Chaohui and Mei, Xue and Prokhorov, Danil and Tao, Dacheng, Multi-Store Tracker (muster): A Cognitive Psychology Inspired Approach to Object Tracking, In Proc. IEEE Int. Conf. Comput. Vis., 2015: 749-758.

[51] Hare, Sam and Golodetz, Stuart and Saffari, Amir and Vineet, Vibhav and Cheng, Ming-Ming and Hicks, Stephen L and Torr, Philip HS, Struck: Structured Output Tracking With Kernels, IEEE Trans. Pattern Anal. Mach. Intell., 2016, 38(10): 2096-2109.

[52] João F. Henriques and Rui Caseiro and Pedro Martins and Jorge P. Batista, Exploiting the Circulant Structure of Tracking-by-Detection With Kernels. In Proc. Eur. Conf. Comput. Vis., 2012: 702-715.

[53] Kalal, Zdenek and Mikolajczyk, Krystian and Matas, Jiri and others, Tracking-Learning-Detection, IEEE Trans. Pattern Anal. Mach. Intell., 2012, 34(7): 1409-1422.

Weiwei Xing received a BS degree in computer science and technology and a PhD in signal and information processing from Beijing Jiaotong University, Beijing, China, in 2001 and 2006, respectively. She is currently a Professor with the School of Software Engineering, Beijing Jiaotong University. Her research interests include intelligent information processing and machine learning.

Yuxiang Yang received a BS degree in computer science and technology from Northeastern University of China, Liaoning, China, in 2014. He is currently a PhD student with the School of Software Engineering, Beijing Jiaotong University. His research interests include image processing, deep learning, reinforcement learning, and target tracking.

Shunli Zhang received BS and MS degrees in electronics and information engineering from Shandong University, Jinan, China, in 2008 and 2011, respectively, and a PhD in signal and information processing from Tsinghua University in 2016. He is currently a faculty member at the School of Software Engineering, Beijing Jiaotong University. His research interests include pattern recognition, computer vision, and image processing.

Liqiang Wang received a PhD in computer science from Stony Brook University in 2006. He is an associate professor in the School of Computer Science at the University of Central Florida. His research interests include big data systems and deep learning. He received a US National Science Foundation CAREER Award in 2011.

Qi Yu received a BS degree in software engineering from the School of Software, North University of China, Shanxi, China, in 2016. She is currently a master student with the School of Software Engineering, Beijing Jiaotong University. Her research interests include data analysis, machine learning, testing, and user experience.