Deep Reinforcement Learning-Based End-to-End Control for UAV Dynamic Target Tracking

Jiang Zhao 1, Han Liu 1, Jiaming Sun 1, Kun Wu 2,*, Zhihao Cai 1, Yan Ma 3 and Yingxun Wang 4

1 School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
2 Flying College, Beihang University, Beijing 100191, China
3 Science and Technology on Information Systems Engineering Laboratory, Beijing Institute of Control & Electronics Technology, Beijing 100038, China
4 Institute of Unmanned System, Beihang University, Beijing 100191, China
* Correspondence: wukun@buaa.edu.cn

Abstract: Uncertainty of target motion, limited perception ability of onboard cameras, and constrained control have brought new challenges to unmanned aerial vehicle (UAV) dynamic target tracking control. In virtue of the powerful fitting ability and learning ability of the neural network, this paper proposes a new deep reinforcement learning (DRL)-based end-to-end control method for UAV dynamic target tracking. Firstly, a DRL-based framework using onboard camera image is established, which simplifies the traditional modularization paradigm. Secondly, neural network architecture, reward functions, and soft actor-critic (SAC)-based speed command perception algorithm are designed to train the policy network. The output of the policy network is denormalized and directly used as speed control command, which realizes the UAV dynamic target tracking. Finally, the feasibility of the proposed end-to-end control method is demonstrated by numerical simulation. The results show that the proposed DRL-based framework is feasible to simplify the traditional modularization paradigm. The UAV can track the dynamic target with rapidly changing of speed and direction.

Keywords: unmanned aerial vehicle (UAV); dynamic target; tracking control; deep reinforcement learning (DRL); end-to-end control; neural network

1. Introduction

With the continuous improvement of its autonomous intelligence level, unmanned aerial vehicle (UAV) is widely used in the civilian field such as aerial photography, agricultural detection [1–5] and in the military field such as aerial reconnaissance, monitoring [6–8]. As a hot and difficult issue, UAV dynamic target tracking technology needs to be solved urgently [9–14]. UAV target tracking control needs to realize the whole process closed-loop of sensing control, which has strong systematic and multidisciplinary characteristics. As for the dynamic target tracking task, the movement form of the target is constantly changing, and the randomness, diversity and complexity presented pose a great challenge to the perception and control system of UAV. Due to the lack of prior knowledge of the motion pattern of the tracked target, how to ensure the UAV respond accurately and quickly to the change of the target uncertainty has become an urgent problem to be solved.

Vision-based target tracking control methods can be divided into two categories: traditional target tracking methods [15–20] and learning-based target tracking methods [13,21–26]. The traditional vision-based UAV target tracking control scheme usually detects the target based on the color, shape and other characteristics, uses the vision-tracking algorithm to estimate the target state according to the image feature points, and then designs the corresponding control law to generate control instructions. The vision-based UAV target tracking control system constructed by [15] is composed of color target detection and tracking algorithm, Kalman filter relative position estimation algorithm and nonlinear controller.
Chakrabarty et al. [16] adopts clustering of static-adaptive correspondences for deformable object tracking (CMT) algorithm [17] to realize the tracking of the target on the image plane, which overcomes the problem of poor effect of open tracking-learning-detection (TLD) algorithm in dealing with object deformation. This method has better robustness to target deformation and temporary occlusion. Based on the optimization of hardware, Greatwood et al. [18] designed a ground target tracking controller which can make full use of the parallel characteristics of images and run on the onboard computer efficiently and in real time, and realized the real-time tracking of quadrotor to the ground vehicle target. Diego et al. [19] introduced a Haar feature classifier to realize the recognition of human targets, and combined with the position of the target in the image and Kalman filter algorithm to realize the position tracking and prediction of moving targets. Petersen et al. [20] proposed a UAV target tracking control architecture composed of a vision front-end, a tracking back-end, a selector and a controller. Based on the improved random sampling consensus algorithm, the feature points between adjacent image frames are converted to obtain a moving target tracker, and the final tracked target is determined after screening by the selector.

For the learning-based methods, the UAV target tracking control scheme inspired by the neural network takes the image as the input and directly outputs the action command through the neural network. Kassab et al. [21] realized a target tracking system through two deep neural networks with the aid of image-based visual servo [22]. The proximity network estimates the relative distance between the UAV and the target based on the results of visual tracking, and the tracking network is used to control the relative azimuth between the UAV and the target. Bhagat [13] proposed an algorithm based on deep reinforcement learning (DRL), which takes the position of UAV, target and obstacles in the environment as input, and selects one action of UAV moving in six directions as output. Li [23] proposed a hierarchical network structure that integrates the perception layer and the control layer into a convolutional neural network to realize autonomous tracking of UAV to human. The input of the network is the monocular image and the state information of the UAV. The output includes the four-dimensional motion vector of the three-axis position offset and the heading angle offset. Zhang [24] proposed a coarse-to-fine scheme with DRL to address the aspect ratio variation in UAV tracking. Zhao [25] proposed an end-to-end cooperative multi-agent reinforcement learning (MARL) scheme, in which the UAV can make intelligent flight decisions for cooperative target tracking according to the past and current state of the target. Xu [26] proposed Multiple Pools Twin Delay Deep Deterministic Policy Gradient (MPTD3) algorithm to complete UAV autonomous obstacle avoidance and target tracking tasks, there are often some problems such as slow convergence speed and low success rate. When the target speed changes rapidly, UAV needs to make timely and accurate response to the change of target motion. With the strong fitting ability and learning ability of neural network, combined with many advantages of DRL, this paper focuses on the research of UAV dynamic target tracking control method based on end-to-end learning.

The main contribution of this paper can be summarized as follows:

1. This paper proposes a DRL-based end-to-end control framework of UAV dynamic target tracking, which simplifies the traditional modularization paradigm by establishing an integrated neural network. This method can achieve dynamic target tracking using the policy obtained from the task training of flying towards a fixed target.
2. Neural network architecture, reward functions, and SAC-based speed command perception are designed to train the policy network for UAV dynamic target tracking. The trained policy network can use the input image to obtain the speed control command as an output, which realizes the UAV dynamic target tracking based on speed command perception.
3. The numerical results show that the proposed framework for simplifying the traditional modularization paradigm is feasible and the end-to-end control method allows the UAV to track the dynamic target with rapidly changing of speed and direction.
The remainder of this paper is organized as follows. In Section 2, the problem formulation is stated, and the preliminaries are introduced. In Section 3, the UAV dynamic target tracking control method is proposed in detail, including the framework, neural network architecture, reward functions and SAC-based speed command perception algorithm. In Section 4, numerical results and discussions are presented. Section 5 summarizes the contribution of this paper and presents future work.

2. Preliminaries

2.1. Problem Formulation

For target tracking problem, the UAV only has prior knowledge about the visual features of the target, but the target motion model is unknown. As Figure 1 shown, the UAV perceives the target and the environment in a limited field of view only through the down-looking monocular camera firmly attached to the bottom of the body. It needs to rely entirely on onboard sensors and onboard computers to process the perception information and generate corresponding control instructions, so that the UAV can track dynamic targets continuously and steadily. To achieve this, an end-to-end control method is proposed to train the UAV to calculate the speed control commands according to the camera image.

![Figure 1. UAV dynamic target tracking problem.](image1.png)

2.2. UAV Model

As shown in Figure 2, the coordinate systems including the camera coordinate system, body coordinate system, pixel coordinate system, world coordinate system and scene coordinate system.

![Figure 2. The definition of coordinate system.](image2.png)
The UAV studied in this paper is X-configuration quadrotor, and it is symmetrically equipped with four motors, whose rotation drives the rotation of the rotator to generate pull to power the UAV. Figure 3 shows the body coordinate system and forces/moments acting on the UAV.

Figure 3. Coordinate systems and forces/moments acting on the UAV.

Assuming that the UAV is a rigid body, it is only subject to gravity in the $O_{bw}z_{bw}$ direction and lift in the $O_{bw}z_{bw}$ negative direction. The position and attitude dynamic models of the UAV are shown in Equations (1) and (2) [27].

\[
mb \ddot{\mathbf{v}}^w = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R^w_b \begin{bmatrix} 0 \\ 0 \\ -(T_1 + T_2 + T_3 + T_4) \end{bmatrix} \tag{1}
\]

where $\mathbf{v}^w = [v_x^w, v_y^w, v_z^w]^T$ represents the velocity of the UAV, $m$ represents the mass of the UAV, $R^w_b$ represents the rotation matrix from body coordinate system to world coordinate system, $T_i (i = 1, 2, 3, 4)$ represents the force generated by the $i$-th motor [28].

\[
J \cdot \dot{\omega}^b = \mathbf{M} \tag{2}
\]

where $J$ represents the inertia of the UAV, $\mathbf{M}$ represents total moment of the UAV.

In the process of the UAV tracking the target, it is assumed that the UAV flies at a fixed altitude and its position is recorded as $(X_{UAV}^w, Y_{UAV}^w, Z_{UAV}^w)^T$. The kinematic model of the UAV is shown in Equation (3).

\[
\begin{bmatrix}
\dot{X}_{UAV}^w \\
\dot{Y}_{UAV}^w \\
\dot{Z}_{UAV}^w
\end{bmatrix} =
\begin{bmatrix}
v_x^w \\
v_y^w \\
v_z^w
\end{bmatrix}.
\begin{bmatrix}
1 & \tan \theta \sin \phi & \tan \theta \cos \phi \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi / \cos \theta & \cos \phi / \cos \theta
\end{bmatrix}
\begin{bmatrix}
p \\
q \\
r
\end{bmatrix} \tag{3}
\]

The target moves in the $O_{bw}X_{bw}Y_{bw}$ plane, and its position is recorded as $(X_T^w, Y_T^w, 0)^T$. The kinematic model of the target is shown in Equation (4).

\[
\begin{bmatrix}
\dot{X}_T^w \\
\dot{Y}_T^w
\end{bmatrix} =
\begin{bmatrix}
v_{xT}^w \\
v_{yT}^w
\end{bmatrix} \tag{4}
\]

In order to describe the relative motion relationship between the UAV and the target, the position vector of the target relative to the UAV in the world coordinate system is defined as $(X_R^w, Y_R^w, Z_R^w)^T = (X_T^w, Y_T^w, Z_T^w)^T - (X_{UAV}^w, Y_{UAV}^w, Z_{UAV}^w)^T$. The position tracking error between the UAV and the target is defined as the difference between the coordinates of the current position of the UAV and the target.
2.3. DRL and SAC

DRL is a cross field of reinforcement learning (RL) and deep neural network. DRL method can perceive complex inputs and make decisions at the same time. Figure 4 shows the basic framework of DRL. This figure can well reflect the interactive characteristics of DRL. At each time step \( t \), the agent interacts with the environment once. The agent is in state \( s_t \) and generates action \( a_t \) according to the policy \( \pi(a_t|s_t; \theta) \) represented by the neural network parameter \( \theta \). After the action acts on the environment, the state of the agent will be updated to \( s_{t+1} \) according to the dynamic model of the environment \( p(s_{t+1}, r_t|s_t, a_t) \). At the same time, the immediate reward \( r_t \) for obtaining environmental feedback is obtained. Therefore, the DRL problem aims to give a series of interaction processes between agent and the environment, and finds the optimal policy to maximize the return \( R_t \).

![Figure 4. Framework of DRL.](image)

Neural network is a mathematical model that simulates the structure of biological neural network and performs distributed parallel information processing. It can adaptively change its own structural parameters based on external information, and store information with the help of parameters such as weight and bias term of each layer. The smallest unit node that constitutes the neural network model is “neuron”, as shown in Figure 5. The output of a neuron in an episode is the result of the activation function after the addition of the weighted sum of the input data and the bias term. The activation function provides the nonlinear expression ability for the neural network, which is differentiable and monotonic [29].

![Figure 5. A single neuron.](image)

The SAC algorithm is an actor-critic DRL algorithm that introduces maximum entropy, in which the actor generates a random policy. The goal of this algorithm is to maximize the cumulative reward regularized by entropy instead of just the cumulative reward. SAC algorithm can increase the randomness of action selection, encourage the agent to explore more in the training process, and thereby speed up subsequent learning to prevent the policy from prematurely converging to a local optimum. Some practical experiments show that SAC algorithm has higher learning efficiency than RL algorithm with traditional objective function.
The basic principle of SAC algorithm is briefly described below. $\pi(a|s;\theta)$ represents the actor network with parameter $\theta$, $Q(s,a;\phi)$ represents Q-network with parameter $\phi$. $V(s;\psi)$ and $\bar{V}(s;\psi)$, respectively, represent the behavior value network and the corresponding target value network. Q-network and V-network together form a critic for the evaluation of the actor network. In each iteration, the agent first interacts with the environment based on the current policy to generate new data and stores it in the experience pool, and then randomly samples from the experience replay buffer and updates the actor, critic and the corresponding target network. Through derivation, the loss function of the V-network is [30]:

$$J_{V}(\psi) = E_{s_t \sim D_s}[\frac{1}{2}(V_{\pi}(s_t;\psi) - E_{a_t \sim \pi(\theta)}[Q_{\pi}(s_t,a_t;\phi)] - \alpha \log \pi(a_t|s_t;\theta))^2]$$

(5)

where $\alpha$ is the temperature parameter that determines the relative importance of the entropy term versus the reward, and thus controls the stochasticity of the optimal policy. The gradient of $V$-network is:

$$\nabla_{\psi} J_{V}(\psi) = \nabla_{\psi} V(s_t;\psi)(V(s_t;\psi) - Q_{\pi}(s_t,a_t;\phi) + \alpha \log \pi(a_t|s_t;\theta))$$

(6)

the loss function of the Q-network is:

$$J_{Q}(\phi) = E_{(s_t,a_t) \sim D_s}[\frac{1}{2}(Q_{\pi}(s_t,a_t;\phi) - \bar{Q}_{\pi}(s_t,a_t))^2]$$

(7)

where, $\bar{Q}_{\pi}(s_t,a_t) = r(s_t,a_t) + \gamma E_{s_{t+1} \sim p}[V(s_{t+1};\bar{\psi})]$. $\gamma$ is the discount factor to ensure that the sum of expected rewards (and entropy) is finite. Then the gradient of Q-network is:

$$\nabla_{\phi} J_{Q}(\phi) = \nabla_{\phi} Q_{\pi}(s_t,a_t;\phi)(Q_{\pi}(s_t,a_t;\phi) - r(s_t,a_t) - \gamma \bar{V}(s_{t+1};\bar{\psi}))$$

(8)

Since actor network generates a random policy, under the setting of continuous action space, re-parameterization is introduced to update the policy to reduce the variance of policy gradient estimation. The policy $\pi(\theta)$ is expressed as a function that uses the state $s$ and the noise vector $\epsilon$ subject to the normal distribution as input to output the action $a$, that is $a = f(s,\epsilon;\theta)$. This process can also be regarded as action sampling from the normal distribution determined by the output of the policy network, and then the loss function of the actor network can be obtained as [31]:

$$J_{\pi}(\theta) = E_{s_t \sim D_s,\epsilon_t \sim N}[\log \pi(f(s_t,\epsilon_t;\theta)|s_t;\theta) - Q_{\pi}(s_t,f(s_t,\epsilon_t;\theta);\phi)]$$

(9)

3. End-to-End Control for UAV Dynamic Target Tracking

3.1. Framework

Based on DRL, we design the end-to-end control method of directly outputting speed control commands from the original images during the interaction between the UAV target tracking agent and the simulation environment. End-to-end control method can directly learn the corresponding control strategy based on high-dimensional sensor input information by establishing a certain structure of depth neural network between the sensing end and the control end, and replacing the manually designed features with automatically extracted hierarchical depth features. The speed control commands can be directly obtained as the input of the subsequent controller through the inverse normalization processing of the network output, to realize the perception control closed loop of the UAV dynamic target tracking. The framework of UAV dynamic target tracking control based on DRL is shown as Figure 6.
The Markov decision process is the most important mathematical model in RL. Therefore, the dynamic target tracking control problem of UAV is first described by Markov decision process, with the emphasis on the definition of its state space and action space.

It can be seen from the analysis of the UAV dynamic target tracking task that the camera image and the state of the UAV at the next moment only depend on the control command generated and executed according to the current image. Therefore, the camera image of the UAV is regarded as an observable state $s_t$, and the control command is regarded as an action $a_t$. The alternation between the two will form a set of state action sequences in time sequence within a finite time domain, recorded as trajectory $\tau = s_0, a_0, s_1, a_1, s_2, a_2, ..., s_{T-1}, a_{T-1}, s_T$, where $s_0$ is the initial state, and $T$ is the termination time of the finite time domain. Figure 7 shows the Markov decision process for UAV dynamic target tracking.

The Markov decision process of UAV target tracking can be described by tuples \( \{S, A, P, R, \gamma\} \): $S$ is the state space. Considering that the original image size of the UAV onboard camera is large, the image after size compression and pixel value normalization to $[0, 1]$ is defined as the state, therefore, $S = \{s \in \mathbb{R}^{120 \times 120 \times 3}\}$. $A$ is the action space. The actions are defined as the desired speed control commands of the UAV normalized in the
horizontal direction, therefore \( A = \{ a = (v_{cmd,x}, v_{cmd,y})^T | v_{cmd,x}, v_{cmd,y} \in [-1, 1] \} \). \( P \) is the state transition function and can be recorded as \( P : S \times A \times S \rightarrow [0, 1] \). The meaning of this function is the probability \( p(s' | s, a) \) that the UAV will acquire the image \( s' \) after sensing the image \( s \) and taking action \( a \). \( R \) is the reward function and can be recorded as \( R : S \times A \rightarrow \mathbb{R} \). \( \gamma \) is discounting factor and can be used to calculate cumulative rewards, \( \gamma \in (0, 1) \).

3.1.2. Interactive Environment and Agents

The environment in the DRL problem refers to the sum of various peripheral elements that interact with the agent, including interface functions to achieve interaction and entities such as targets and surrounding scenes. The interface functions used for UAV target tracking task mainly include initialization function, reset function and single step interaction function.

The initialization function is used to declare and initialize the parameters of the environment and some global variables shared by multiple functions, including the altitude of the UAV flying at fixed altitude, the starting position of the target, the boundary of the camera field of view, and the number of interaction steps. The reset function is used to reset the UAV, target and global variables in the environment when the agent triggers the episode termination condition, and returns the image observed by the UAV at the reset position as the initial state of the new episode. The single-step interaction function is used to make the agent interact with the environment once in each training step. The normalized action generated by the agent at the current time is taken as the input, and after the speed control command is restored and limited, it is forwarded to the UAV model for execution. Next, the physical state of the UAV is updated at a certain interval, and the new image in the field of view can be obtained as the new state to which the interaction is transferred. Then, the reward function calculates the single-step reward and determines whether the episode termination conditions are met. Finally, the function returns the normalized new state, the single-step reward, the episode end flag amount, and the related annotation information.

The agent in DRL has the ability of action decision-making and self-renewal, and the core is the policy itself approximated by the deep neural network, that is \( \pi : S \times A \rightarrow [0, 1] \), represents the transition probability from state to action. Since the state space of the UAV target tracking control problem is a high-dimensional space, in the design of the policy network, the design idea of multiple feature extraction network is adopted. The multilayer convolutional neural network is selected as the first half of the policy network \( \pi(a | s; \theta) \), and a hidden layer with spatial feature extraction function is added before the full connection layer of the second half to enhance the expression of the position information of the target in the image. Considering the stability and convergence effect of neural network training, it is also necessary to normalize the state input of the agent policy.

The use of agents has two modes: training and testing. The agent is designed with actor-critic framework, so it needs to maintain both actor and critic networks during the training process, but only needs to run actor network during the testing process. In the training mode, the agent uses the collected interactive data to iteratively update its policy network parameters according to certain rules. The policy gradually converges to the vicinity of the optimal policy, making the action output generated according to the state input more and more ideal and accurate. In the testing mode, the trained agent policy network is loaded and its parameters and structure are fixed. The agent only needs to perform forward propagation calculation according to the incoming image state \( s_t \) at each time step to obtain the action output \( a_t \).

3.2. Neural Network Architecture for End-to-End Learning

The designed actor network architecture and critic network architecture for end-to-end learning are shown in Figure 8. Their backbone networks are both composed of three convolution layers and spatial index normalization layers.
Figure 8. The neural network architecture: (a) the architecture of actor-network; (b) the architecture of critic-network.

The actor network has two branches at the last output layer, which are, respectively, used to calculate the mean and logarithmic variance of the generated random action Gaussian distribution. The critic network is behind the backbone network. For the Q-network, the upper layer feature vector should be spliced with the current action vector as the input of the subsequent fully connected layer, and the final output is a scalar, that is, the estimated Q value; while for the V-network, the action vector is not required as an additional input. The feature vector output by the spatial index normalization layer can be used as the input of the subsequent fully connected layer, and the final output is the V value.

3.3. Reward Function for Target Tracking

The designed reward function is mainly composed of three items. The first item $r_1$ is related to the relative distance between the UAV and the target in the horizontal direction at the current time. The second item $r_2$ is related to the action direction calculated by the agent at the current time and the relative orientation between the UAV and the target, and the third item $r_3$ is related to the episode termination condition.

The design of $r_1$ is to encourage the action of the UAV to approach the target and punish the action of the UAV to move away from the target. $d_r$ represents the distance between the UAV and target in the current step; $d_{r,\text{last}}$ represents the distance between the UAV and target in the last step; $d_{\text{lim}}$ represents the limit distance that the UAV can track the target. When $d_r = d_{\text{lim}}, d_{r,\text{last}} > d_r$, we set $r_1 = 0$. The closer the UAV is to the target in the process of approaching the target, the greater the positive reward value obtained; the farther the UAV is from the target in the process of moving away from the target, the greater the absolute value of the negative reward obtained. Since the reward in an episode is the accumulated reward value for a period of time, when the UAV changes from approach the target in the previous step to moving away from the target in the current step, the absolute value of the punishment in the current step is greater than the reward in the previous step. When the UAV approaches the target for several consecutive steps, it is weighted by the number of consecutive steps. $r_1$ is calculated as follows:

$$r_1 = \begin{cases} (-5d_r + 5d_{\text{lim}}) \cdot n_{\text{approach}}, & \text{if } d_{r,\text{last}} > d_r \\ 5d_{r,\text{last}} - 5d_{\text{lim}} - 2d_r, & \text{if } d_{r,\text{last}} < d_r \end{cases}$$

(10)

where approach step $n_{\text{approach}}$ is cleared when the UAV is far from the target, and accumulated by 1 when the UAV approaches the target.

The design of $r_2$ is to reward and punish the action direction. Firstly, the actual azimuth angle of the target relative to the UAV is calculated based on the current position of the UAV and the target. Secondly, the included angle $\theta_{\text{error}}$ between the action direction vector $a$ and the actual relative azimuth direction vector $a_0$, is calculated by using the cosine
Theorem. If the included angle is less than a threshold $\theta_{\text{thresh}}$, a positive reward inversely proportional to the included angle is given; otherwise, set this item as negative reward and the greater the included angle, the greater the absolute value of the negative reward. In addition, in order to prevent an excessive value when $\theta_{\text{thresh}}$ approaching 0, it is necessary to limit it when $r_2$ is a positive reward. $r_2$ is calculated as follows:

$$
\theta_{\text{error}} = \arccos\left( \frac{\mathbf{a} \cdot \mathbf{a}_{ij}}{\left| \mathbf{a} \right| \left| \mathbf{a}_{ij} \right|} \right), \left| \mathbf{a}_{ij} \right| = d_r
$$

(11)

$$
r_2 = \begin{cases} 
\frac{1}{2\theta_{\text{error}}}, & \text{if } \theta_{\text{error}} < \theta_{\text{thresh}} \\
-2\theta_{\text{error}}, & \text{else}
\end{cases}
$$

(12)

In the design of $r_3$, the judgment of episode termination conditions is mainly considered. Assuming three conditions to trigger the termination of the episode: the failure of the episode mission caused by the loss of the target in the field of vision, the success of the episode mission caused by the UAV moving to a certain area directly above the target and meeting certain conditions, and the maximum number of steps reached in the episode. The reward function is only set for the first two conditions in this paper. When the relative distance between the UAV and the target are greater than the geographical boundary $d_{\text{lim}}$ constrained by the camera’s field of vision, it is deemed that the mission of this episode has failed. The UAV is directly given a negative reward $r_{\text{out}}$ with a large absolute value and the influence of the other two rewards is shielded. When the horizontal distance between the UAV and the target is less than a certain threshold $d_r$, it is considered that the UAV has successfully completed the task. At this time, a positive reward $r_{\text{success}}$ weighted by the times of consecutive successes $n_{\text{success}}$ is added based on the first two rewards. The times of successes $n_{\text{success}}$ only counts the number of consecutive steps that meet the UAV’s threshold above the target.

$$
r_3 = \begin{cases} 
\text{in success}, & \text{if } d_r > d_{\text{lim}} \\
\text{out}, & \text{else}
\end{cases}
$$

(13)

In conclusion, the reward function designed in this paper for the DRL problem of UAV target tracking has the following form:

$$
r = \begin{cases} 
 r_3, & \text{if } d_r > d_{\text{lim}} \\
 w_1 \cdot r_1 + w_2 \cdot r_2 + r_3, & \text{else}
\end{cases}
$$

(14)

where $w_1, w_2$ are the corresponding weight coefficients.

3.4. SAC-Based Speed Command Perception

To reduce the deviation of Q value calculation, two Q-networks are maintained during actual training, and the smaller Q value among the outputs of the two Q-networks are used to calculate the loss of the policy network and the V-network [32]. Note that two Q-networks are $Q_1(s, a; \phi_1)$ and $Q_2(s, a; \phi_2)$, respectively, then the loss function of the V-network is:

$$
J_V(\psi) = E_{s_t \sim D}\left[ \frac{1}{2} \left( V_{\pi}(s_t; \psi) - E_{a_t \sim \pi(\theta)}[\min_{i=1,2} Q_i(s_t, a_t; \phi_i)] - \alpha \log \pi(a_t|s_t; \theta) \right)^2 \right]
$$

(15)

where $\tilde{\pi}(\cdot)$ is also obtained based on action sampling, that is, $\tilde{\pi}_t = f(s_t, \epsilon_t; \theta)$.

Accordingly, the loss function of the policy network is:

$$
J_\pi(\theta) = E_{s_t \sim D_s, \epsilon_t \sim N} \left[ \log \pi(f(s_t, \epsilon_t; \theta)|s_t; \theta) - \min_{i=1,2} Q_i(s_t, f(s_t, \epsilon_t; \theta); \phi_i) \right]
$$

(16)

Then, end-to-end learning for speed command perception training framework based on SAC algorithm can be obtained, as shown in Figure 9.
The SAC-based training algorithm is summarized as Algorithm 1. In practical application, the agent only needs to load the neural network model obtained through the above training process, and perform inverse normalization processing on the network output to generate speed control commands for interaction with the environment.

Algorithm 1. SAC-Based Training Algorithm

1. Initialize the learning rate of each network $\lambda_V, \lambda_Q, \lambda_\pi$;
2. Initialize the target network soft update rate $\tau$, entropy regularization weights $\alpha$;
3. Initialize each network parameter $\theta, \phi, \psi, \overline{\psi}$;
4. Initialize the replay buffer;
5. For each episode:
   i. Take the current state as the input of the current actor network, and generate actions $a_t$;
   ii. Normalize the actions and convert them into speed control commands;
   iii. Control the UAV with control commands and observe reward $r_{t+1}$ and observe new image state $s_{t+1}$;
   iv. Store the piece of experience $(s_t, a_t, s_{t+1}, r_{t+1})$ into the replay buffer;
   v. Sample replayed experience $(s_t, a_t, s_{t'}, r_{t'})$ from the replay buffer;
   vi. Update the behavior value network: $\psi \leftarrow \psi - \lambda_\pi \nabla_\phi J_\psi(\psi)$;
   vii. Update the Q network: $\phi \leftarrow \phi - \lambda_Q \nabla_\phi J_Q(\phi)$;
   viii. Update the policy network: $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$;
   ix. Update the target value network: $\overline{\psi} \leftarrow \tau \overline{\psi} + (1 - \tau)\psi$;
   x. If the terminal condition is satisfied, start a new episode. Or, continue for next time step.

The end of a time step;
The end of an episode;
4. Numerical Simulations

In this section, training simulation and three UAV dynamic target tracking simulations with different conditions are executed to test the performance of the trained policy network.

4.1. Training Results

In this subsection, the policy network is trained by the proposed method. In order to test the task completion ability of the obtained policy network, 100 random start point tests are conducted using the simulation scenario including the target fixed at \((0, 0, 0)\) m during the training.

As for the training parameters setting, the size of experience replay buffer is 10,000. The size of experience replay batch is 128. The discounting factor is 0.99. The maximum number of steps each episode is 50. The learning rate of each neural network is 0.0003, as shown in Table 1.

Table 1. The parameters of training simulation.

| Training Parameters                          | Value  |
|---------------------------------------------|--------|
| Size of experience replay buffer            | 10,000 |
| Size of experience replay batch             | 128    |
| Discounting factor                          | 0.99   |
| Maximum number of steps each episode        | 50     |
| Learning rate of each neural network        | 0.0003 |

After about 40,000 steps of interaction, the cumulative reward variation curve of the episode is shown in Figure 10a. It can be seen that it rises rapidly at about 5000 steps, and then maintains a large positive level, which indicates that the agent has learned the policy of obtaining a large cumulative reward through interaction with the environment, which reflects the effectiveness of the end-to-end learning process of speed command perception. For DRL agents, the success rate is generally used as an indicator to measure the quality of training results. The trajectories of the UAV during the testing are shown in Figure 10b. The UAV can fly from any starting position to the top of the fixed target, and the mission success rate is 100%. It shows that the obtained policy network is feasible to complete the equivalent task of UAV dynamic target tracking. In the following simulation experiments, the policy network is directly used to track the dynamic target. One of the evaluation loss results of actor and critic network from our many trainings are shown in Figure 10c,d.

4.2. Dynamic Target Tracking

In this subsection, the target tracking effect of UAV is tested under three conditions that the target moves along different trajectories, as shown in Table 2. The altitude of UAV fixed altitude flight is set to 5 m.

Table 2. The initial conditions of the three different cases.

| Target Trajectory      | Target Velocity |
|------------------------|-----------------|
| Case 1                 | Square trajectory 0.5 m/s |
| Case 2                 | Polygonal trajectory 0.5 m/s |
| Case 3                 | Curve trajectory 0.5~1.2 m/s |

Detailed description and analysis of the simulation results under each condition are given in the following.
4.2.1. Case 1: Square Trajectory

For the first case, the target starts at (0,0,0) m and moves along a square trajectory with a side length of 8 m. The target makes a uniform linear motion on the side of the square, and when it moves to the vertex of the square, the velocity direction will change by 90° and then continue to move in a uniform linear motion in the new direction. The UAV hovers directly above the target at the start time.

Figure 11 shows the simulation testing results. In summary, the UAV can complete the stable tracking of the dynamic target along the square trajectory. The UAV has no overshoot in the process of tracking the target, and the average distance tracking error between the UAV and the target is 0.75 m. The tracking effect of the UAV in the x-axis direction and the y-axis direction is different. The absolute value of the maximum position tracking error in the x-axis direction is 0.80 m, but in the y-axis direction is about 1.14 m. This situation may be caused by the uneven distribution of interaction data used in training in all directions. The velocity of UAV can keep stable near the target velocity, and rarely exceeds the target velocity, which indicates that the velocity control command directly generated by the neural network is conservative.

Figure 10. The training results of UAV target tracking. (a) The variation curve of episode cumulative reward in training process of speed command perception; (b) the position of UAV and target; (c) evaluation policy loss result of in training process; (d) evaluation value loss results of in training process.
This situation may be caused by the uneven distribution of interaction data used in training in all directions. The velocity of UAV can keep stable near the target velocity, and rarely exceeds the target velocity, which indicates that the velocity control command directly generated by the neural network is conservative.

4.2.2. Case 2: Polygonal Trajectory

For the second case, the target starts at \((0, 0, 0)\) m and moves along a polygonal trajectory. Specifically, after the target moves along the straight line \(y = x\) for a certain distance, the sudden change of \(-135^\circ\) occurs in the velocity direction and moves along the negative direction of the \(y\)-axis for a certain distance, and then the sudden change of \(+135^\circ\) occurs in the velocity direction. After repeating this for several times, the target stops moving. The UAV hovers directly above the target at the start time.

Figure 12 shows the simulation testing results. The tracking effect of the UAV in the \(x\)-axis direction where the target motion change is relatively gentle is acceptable and the absolute value of the maximum position tracking error in this direction is about 0.57 m. However, the tracking hysteresis in the \(y\)-axis direction where the target motion changes greatly is obvious and the absolute value of the maximum position tracking error in the \(y\)-axis direction is about 1.14 m. The average distance tracking error between UAV and target is about 0.61 m.
4.2.3. Case 3: Curve Trajectory

For the third case, the target starts at (0, 0, 0) m and moves along a lemniscate curve trajectory. The moving velocity of the target is slower at the place where the curvature of the lemniscate is small and faster at the place where the curvature is large, and generally changes within the range of 0.5 m/s~1.2 m/s. The UAV hovers directly above the target at the start time.

Figure 13 shows the simulation testing results. In summary, the motion trajectory of UAV and target fit well. The average value of the distance tracking error between the UAV and the target is 1.04 m. The absolute value of the maximum position tracking error in the x-axis direction is about 1.18 m, and in the y-axis direction is about 1.49 m. The velocity of the UAV fluctuates at the local maximum of the target velocity, which indicates that the tracking is difficult when the target velocity changes greatly.

Figure 12. The state curves of UAV and target under case 2. (a) 3D trajectory; (b) 2D trajectory in the x–y plane; (c) x–axis position; (d) y–axis position; (e) position tracking error; (f) distance tracking error; (g) component velocity; (h) resultant velocity; (i) control commands for the actuators.
Figure 13. The state curves of UAV and target under case 3. (a) 3D trajectory; (b) 2D trajectory in the \(x-y\) plane; (c) \(x\)-axis position; (d) \(y\)-axis position; (e) position tracking error; (f) distance tracking error; (g) component velocity; (h) resultant velocity; (i) control commands for the actuators.

4.3. Discussion

The numerical simulation results under different cases demonstrate the effectiveness of the dynamic target tracking control method of UAV based on DRL. Through three groups of simulation tests, we examine the performance of UAV when tracking target with constant speed, target with sudden change in direction, and target with changing speed and direction.

In the first two groups of simulations, the directions of the target speed changes are \(90^\circ\) and \(135^\circ\), respectively. In the first simulation, the component speed of UAV can keep 0.5 m/s. While in the second simulation, the component speed of UAV is only 0.33 m/s. Therefore, the greater the change of target speed direction, the more difficult for UAV dynamic target tracking. In the third simulation, the speed and direction of target concurrently change, and the distance error of this simulation is 1.04 m, larger than the other two simulations. All the simulation results show that the UAV can complete the target tracking task under various conditions.
5. Conclusions

This paper proposes a new DRL-based end-to-end control method for UAV dynamic target tracking. This method is demonstrated based on several numerical simulation experiments. The UAV can track target with sudden changes of 90° or 135° in direction and target with speed varying from 0.5 m/s to 1.2 m/s. In addition, the SAC-based algorithm can accelerate the subsequent learning speed and prevent the policy from converging to the local optimal value prematurely. The end-to-end control using neural networks can be used for obstacle avoidance, landing control, et al. For future work, the model trained by neural network method will be used in flight experiment to demonstrate the feasibility of the end-to-end control method. A comparison between the proposed method with the traditional methods will also be performed.

Author Contributions: Methodology, J.Z., H.L. and J.S.; validation, H.L. and J.S.; resources, K.W., Z.C., Y.M. and Y.W.; writing—original draft preparation, J.Z. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Fundamental Research Funds for the Central Universities of China (No.YWF-22-L-539).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Kondoyanni, M.; Loukatos, D.; Maraveas, C. Bio-Inspired Robots and Structures toward Fostering the Modernization of Agriculture. Biomimetics 2022, 7, 69. [CrossRef] [PubMed]
2. Mademlis, I.; Nikolaidis, N.; Tefas, A. Autonomous UAV cinematography: A tutorial and a formalized shot-type taxonomy. ACM Comput. Surv. 2019, 52, 1–33. [CrossRef]
3. Birk, A.; Wiggerich, B.; Bülow, H. Safety, security, and rescue missions with an unmanned aerial vehicle. J. Intell. Robot. Syst. 2011, 64, 57–76. [CrossRef]
4. Radoglou-Grammatikis, P.; Sarigiannidis, P.; Lagkas, T. A compilation of UAV applications for precision agriculture. Comput. Netw. 2020, 172, 107148. [CrossRef]
5. Messina, G.; Modica, G. Applications of UAV thermal imagery in precision agriculture: State of the art and future research outlook. Remote Sens. 2020, 12, 1491. [CrossRef]
6. Gu, J.; Su, T.; Wang, Q. Multiple moving targets surveillance based on a cooperative network for multi-UAV. IEEE Commun. Mag. 2018, 56, 82–89. [CrossRef]
7. Zhao, J.; Xiao, G.; Zhang, X. A survey on object tracking in aerial surveillance. In Proceedings of the International Conference on Aerospace System Science and Engineering, Berlin, Germany, 31 July 2018; pp. 53–68.
8. Chamola, V.; Kotesh, P.; Agarwal, A.; Gupta, N.; Guizani, M. A comprehensive review of unmanned aerial vehicle attacks and neutralization techniques. Ad Hoc Netw. 2021, 111, 102324. [CrossRef] [PubMed]
9. Bhagat, S.; Sujit, P.B. UAV Target Tracking in Urban Environments Using Deep Reinforcement Learning. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020; pp. 694–701.
10. Wang, S.; Jiang, F.; Zhang, B. Development of UAV-based target tracking and recognition systems. IEEE Trans. Intell. Transp. Syst. 2019, 21, 3409–3421. [CrossRef]
11. Azrad, S.; Kendoul, F.; Nonami, K. Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. J. Syst. Des. Dyn. 2010, 4, 255–268. [CrossRef]
12. Chakrabarty, A.; Morris, R.; Bouyssounouse, X. Autonomous indoor object tracking with the Parrot AR. Drone. In Proceedings of the International Conference on Unmanned Aircraft Systems, Arlington, TX, USA, 7–10 June 2016; pp. 25–30.
17. Nebehay, G.; Pflugfelder, R. Clustering of static-adaptive correspondences for deformable object tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 6–8 June 2015; pp. 2784–2791.

18. Greatwood, C.; Bose, L.; Richardson, T. Tracking control of a UAV with a parallel visual processor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, BC, Canada, 24–28 September 2017; pp. 4248–4254.

19. Diego, A.; Mercado, R.; Pedro, C. Visual detection and tracking with UAVs, following a mobile object. Adv. Robot. 2019, 33, 388–402.

20. Petersen, M.; Samuelson, C.; Beard, R. Target tracking and following from a multirotor UAV. Curr. Robot. Rep. 2021, 2, 285–295. [CrossRef]

21. Kassab, M.A.; Maher, A.; Elkazzaz, F. UAV target tracking by detection via deep neural networks. In Proceedings of the IEEE International Conference on Multimedia and Expo, Shanghai, China, 8–12 July 2019; pp. 139–144.

22. Shaferman, V.; Shima, T. Cooperative uav tracking under urban occlusions and airspace limitations. In Proceedings of the AIAA Guidance Navigation and Control Conference and Exhibit, Honolulu, HI, USA, 18–21 August 2008; p. 7136.

23. Li, S.; Liu, T.; Zhang, C. Learning unmanned aerial vehicle control for autonomous target following. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 10–15 July 2018; pp. 4936–4942.

24. Zhang, W.; Song, K.; Rong, X.; Li, Y. Coarse-to-fine UAV target tracking with deep reinforcement learning. IEEE Trans. Autom. Sci. Eng. 2018, 16, 1522–1530. [CrossRef]

25. Xia, Z.; Du, J.; Wang, J.; Jiang, C.; Ren, Y.; Li, G.; Han, Z. Multi-Agent Reinforcement Learning Aided Intelligent UAV Swarm for Target Tracking. IEEE Trans. Veh. Technol. 2021, 71, 931–945.

26. Xu, G.; Jiang, W.; Wang, Z.; Wang, Y. Autonomous Obstacle Avoidance and Target Tracking of UAV Based on Deep Reinforcement Learning. J. Intell. Robot. Syst. 2022, 104, 60. [CrossRef]

27. Quan, Q. Introduction to Multicopter Design and Control; Springer: Singapore, 2017; pp. 80–150.

28. Li, M.; Cai, Z.; Zhao, J. Disturbance rejection and high dynamic quadrotor control based on reinforcement learning and supervised learning. Neural Comput. Appl. 2022, 34, 11141–11161. [CrossRef]

29. Huang, H.; Ho, D.W.; Lam, J. Stochastic stability analysis of fuzzy Hopfield neural networks with time-varying delays. IEEE Trans. Circuits Syst. II Express Briefs 2005, 52, 251–255. [CrossRef]

30. Haarnoja, T.; Zhou, A.; Hartikainen, K. Soft actor-critic algorithms and applications. arXiv 2018, arXiv:1812.05905.

31. Haarnoja, T.; Zhou, A.; Abbeel, P. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10-15 July 2018; pp. 1861–1870.

32. Van, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; p. 30.