Received 9 August 2023, accepted 6 October 2023, date of publication 16 October 2023, date of current version 9 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325062

RESEARCH ARTICLE

Deep Reinforcement Learning Tf-Agent-Based Object Tracking With Virtual Autonomous Drone in a Game Engine

KHURSHEDJON FARKHODOV[1], **SUK-HWAN LEE**[2], **JAN PLATOS**[3], **(Member, IEEE), AND KI-RYONG KWON**[1]
[1]Department of AI Convergence, Pukyong National University, Busan 48513, South Korea
[2]Department of Computer Engineering, Dong-A University, Busan 49315, South Korea
[3]Department of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava, 708 00 Ostrava, Czech Republic

Corresponding author: Ki-Ryong Kwon (kiryongkwon@gmail.com)

This work was supported in part by the Ministry of Science and Information and Communication Technology (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program, Supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP), under Grant IITP-2023-2020-0-01797; and in part by the MSIT through the ICT Consilience Creative Program, Supervised by the IITP, under Grant IITP-2023-2016-0-00318.

ABSTRACT The recent development of object-tracking frameworks has affected the performance of many manufacturing and industrial services such as product delivery, autonomous driving systems, security systems, military, transportation and retailing industries, smart cities, healthcare systems, agriculture, etc. Achieving accurate results in physical environments and conditions remains quite challenging for the actual object-tracking. However, the process can be experimented with using simulation techniques or platforms to evaluate and check the model's performance under different simulation conditions and weather changes. This paper presents one of the target tracking approaches based on the reinforcement learning technique integrated with TensorFlow-Agent (tf-agent) to accomplish the tracking process in the Unreal Game Engine simulation platform AirSim Blocks. The productivity of these platforms can be seen while experimenting in virtual-reality conditions with virtual drone agents and performing fine-tuning to achieve the best or desired performance. In this paper, the tf-agent drone learns how to track an object integration with a deep reinforcement learning process to control the actions, states, and tracking by receiving sequential frames from a simple Blocks environment. The tf-agent model is trained in the AirSim Blocks environment for adaptation to the environment and existing objects in a simulation environment for further testing and evaluation regarding the accuracy of tracking and speed. We tested and compared two approaches, DQN and PPO trackers, and reported results in terms of stability, rewards, and numerical performance.

INDEX TERMS Object tracking, object detection, reinforcement learning, AirSim, virtual environment, virtual simulation, tf-agent, unreal game engine.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAV) are widely utilized in every field of manufacturing and daily service procedures [1], where drones help to make the process easier by automating and decreasing the consumption time with safer service activities. For example, in recent years, drones have become a top priority for technical assistance in the agricultural field for the medication and fertilization processes. Additionally, drones can observe cultivated field mapping properly [2] with important information about any irregular changes in the grower's field. In most cases, it can be beneficial for increasing the productivity of large plantations with congenerous growers to determine drainage patterns and wet-dry spots of field elevation that allow more efficient watering techniques. Moreover, there are several fields in which drones are being applied such as surveillance and control of large unreachable areas [3], medical purposes such as Automated

The associate editor coordinating the review of this manuscript and approving it for publication was Gangyi Jiang.

Emergency Drones (AED) [4], search and rescue operations [5], road traffic monitoring applications [6], weather sensing in an urban environment application [7], emergency system applications for firefighter processes [8], and entertainment (photography and cinematography) [9]. Many recent innovations and technologies related to UAV frameworks include surveillance and target tracking units for several reasons, including security maintenance, control, military assistance, and traffic monitoring, besides manufacturing optimization and control. The current development of innovative technologies is taking novel ideas from modeling systems such as virtual reality environment simulators. Creating a virtual version of the physical objects and process simulations can provide proper service optimization and allow for free experimentation with conditions for any activity.

A. IMPORTANCE AND DEVELOPMENT OF OBJECT TRACKING WITH MACHINE LEARNING

In recent years, most of the UAV-based research community has paid attention to improving the performance of visual tracking techniques with several neural networks related to network-based architectures, such as CNN [10], DNN [11], LSTM [12], RL [13], and others. However, these research works present training and testing results in physical environment datasets, such as image collections taken from different cameras in public areas, and image/video sets taken by drone cameras. These object-tracking applications work with certain object classes where the drone tracks dynamic objects in an exact pathway and localizes objects with specific methods as an additional task for the target-tracking framework. Still, there are challenges while tracking moving and static objects with apparently identical aspects to recognize which one is an actual tracking target. Recent object-tracking research development has begun with intensive learning with virtual reality integration platforms [14], [15], so scientists can integrate their technique or algorithm with a virtual reality platform to test their proposals with various fine-tuning parameters and conditions. It gives scholars more opportunities to explore their methods better and more deeply in a hardware-free environment at zero cost and optimize them as much as possible. Additionally, there are some techniques motivated by robust extension of integral schemes for mismatched uncertain nonlinear systems proposed to support asymptotic tracking [16]. Asymptotic tracking means ensuring that the systems' output tracks a desired reference trajectory over time with negligible tracking error. The main goal is to design a tracking control system that guarantees the output converges to the reference trajectory as time approaches infinity in uncertain environments. Another model is the output feedback adaptive rise control technique [17] used for uncertain nonlinear systems to achieve accurate tracking of desired trajectories. The term ''adaptive'' indicates that the controller parameters are updated online based on the system's behavior and the tracking error. A deep Q-learning-based [18] approach has been suggested for firefighting situations, which is obtainable in some agent robots or drones for finding or planning paths and navigating through fire environments. In such complex and hazardous cases, it is required to be more careful to control the situation with concrete plans and actions for rescuing injured or victims of the incident by coordinating situational awareness with other rescuers, which is an urgent task. When the framework is installed and applied to real drones or robots, it can ensure firefighters or rescuers make the right decision in extreme, panicky, and disorienting conditions.

B. DESCRIPTION OF THE VR-BASED WORK AND MAIN CONTRIBUTION

Several relevant research studies that integrate virtual simulation platforms with proposal algorithms have been published. Kalidas et al. [19] presented vision-based navigation of UAVs based simply on image data by employing deep reinforcement learning to avoid stationary and movable obstacles autonomously in discrete and continuous action space. W. Zhao et al. [20] also proposed a perception-based hierarchical active tracking control for UAVs deploying a high-level controller and action orders in a V-REP-based environment. A trained PPO algorithm [21] with reward shaping for aircraft direction to a moving destination in a three-dimensional continuous space model was suggested, with the agent-specific target guidance in virtual state space using a novel reward calculation. Using a PPO-based DRL algorithm [22] was suggested for UAV tracking with the assistance of another UAV, introducing the generalized distributed deep reinforcement learning platform, which provides solutions to overcome various problems such as tracking, controlling, and mission coordination of UAVs. Moreover, M. A. B. Abdelkader et al. [23] propose RL-based drone elevation control on a Python-Unity integrated simulation framework to achieve a stable user diagram protocol (UDP) with the suggested algorithm. Çetin [24] proposes counting drones in a 3D space with several DRL methods present to count drones with another drone in the environment provided by an AirSim simulator.

In this study, we developed an algorithm based on tf-agent drone tracking in a Blocks environment where the tf-agent actively makes decisions to track the target object in the runtime environment. The proposed method includes different reward techniques to boost the learning, tracking, and decision-making processes via a TF-agent-based drone in a simulation platform. There is some computational consideration for correctly applying parameter values to achieve a higher accuracy rate, and state representation was formulated to clear out unnecessary losses and constraints for the training and testing processes.

The primary contributions of this paper can be summarized as follows:

- We introduce a virtual environmental simulation-based object-tracking algorithm model that receives input images directly from a realistic virtual platform.
- Direct access to the network feedable source images from the simulation environment provides the framework

the more advantages of learning and testing when it comes to unknown environmental conditions.

- The experiment is implemented in an AirSim-based basic Blocks environment with a random, particular walking person being tracked by a virtual drone agent.
- Two different methods were adopted and integrated with the virtual simulation platform to demonstrate the performance of the models.

II. RELATED WORKS

The recent development of object tracking via reinforcement learning has improved by integrating it with many target tracking techniques, which produce better performance with decision-making in tracking procedures. Although most visual tracking concepts based on DRL could perform better in the case of the representation model with adopted manners for locating the target object within a search region, the final estimated target coordinates are ideally centered.

A. OBJECT TRACKING VIA DEEP REINFORCEMENT LEARNING

The advancement of object tracking via reinforcement learning is a comparatively novel idea, where object localization and tracking integration with a decision-making model [13], [25], [26], [27], [28], [29], [30] are applied to the learning and tracking process as well. Several studies have discovered that combining deep and RL [31] in various settings confers many advantages. Visual object tracking [32], localizing temporal activity [33], identifying object classes [26], object recognition through video sequence [34], and segmentation [35] are just a few of the computer vision problems that have used DRL. Notably, visual object tracking via DRL framework studies has increased in recent years, where the DRL was associated with several techniques to robust the training and decision-making ability while targeting object location. The agent must estimate the target position (bounding box) in every sequence frame in the most typical use of DRL on visual object tracking by repeatedly selecting ultimate fitting actions to get accurate tracking results.

Accordingly, the state representation is the fulfillment status of the general frame states within a targeted bounding box. In general, actions are the transformation result of the bounding box while tracking that can shift, scale, and turn actions depending on how the network learned and adapted to the environment in training time. In DRL-based object tracking, accuracy (precision) is emphasized as a reward value, showing the difference between the targeted action bounding box and ground truth values. In general, it is called intersection-over-union (IoU). Reward values will change according to the action value difference with ground truth output, which shows tracking accuracy.

B. VIRTUAL SIMULATION-BASED OBJECT TRACKING VIA DEEP REINFORCEMENT LEARNING

In the last few years, most research topics have interacted with innovative trends in virtual simulation world environments that allow the simulation of any action, object, or process, enabling experimentation with complex conditions to manage and optimize results. Algorithm integration with simulation platforms makes it challenging to conduct testing and experimentation while taking advantage of simulation behavior closely related to real-world models with dynamic and inactive action modes. Several simulation platforms have flexible functionality to connect with software algorithms for experimentation. The most widely used open-source platforms currently are AirSim [14] and Unity [15], which intend to bridge the gap between the virtual and real worlds to support the development of autonomous control and a realistic replica of the actual world. Both platforms are advancing their technical abilities with high intensity to positively influence the development and testing of data-driven machine intelligence techniques such as reinforcement learning and deep learning. W. Luo et al. suggested an active object tracking technique [29] via deep reinforcement learning, in which a drone agent adopted the ConvNet-LSTM function approximator for predicting the target movement using a frame-to-action strategy. Besides, they perform additional (ViZDoom and Unreal Engine simulation) environment augmentation techniques and a customized reward function to boost the training process to achieve better target tracking performance. Another virtual simulation-based approach [36] uses a monocular onboard camera via a DRL model to follow the detected target object. They state that this technique is a more accurate and cost-efficient strategy for adopting an algorithm in a virtual environment by using multiple sensor data points from the pre-calculated trajectory. The proposed model combines one of the object detection models called MobileNet [37] to get the bounding box information from the image input of the learning process. The model includes convergence-based exploration and exploitation for adaptively aligning algorithms with the network.

Moreover, J. Schulman et al. suggested a reinforcement learning-based drone follow-me behavior object tracking framework [38] using the Deep Q-Learning (DQN) model to control RL agents with adaptive and flexible behavior. In this object-tracking model, stacked image frames and the inclusion of depth information to integrated as input frames to the learning and testing process. The proposed model has experimented with the different level environments with several structural changes reasonably. Experimental output with several specific conditions showed that the RL-based drone following technique succeeded in its adaptive and generalizing behavior.

In our recent research, we proposed virtual simulation-based visual object tracking via a deep reinforcement learning algorithm [25], which the AirSim drone agent uses to track the targeted object class in a runtime virtual simulation environment by utilizing sequential frames directly from it. Additionally, the suggested model has been tested with a public dataset to evaluate the performance of recent research outputs. The main advantage of a virtual simulation platform is that researchers can conduct experimentation several times with different fine-tuning techniques at no cost until they improve their proposal with high accuracy. Accordingly, generating new, fake, or augmented data or collecting or reusing data from public sets to reinforce model learning and boost localization exactness while decreasing estimation time and human effort is unnecessary.

III. PROPOSED METHOD

In this technique, we created an algorithm implementation that includes several components of the object tracking framework, including training and tracking, and we evaluated it in a virtual simulation environment. The framework's fundamental idea is to learn the action space using direct input from a virtual simulation platform. Platforms allow training the network with little effort spent actively learning and tracking operations. However, the method must be correctly linked with the Q-learning network model to get the required object feature information to study the environment and aid in making continuous action decisions in each frame of the tracking sequence.

A. ALGORITHM BASELINE

Figure 1 shows the baseline of our proposed method illustration. Firstly, the AirSim simulation platform must be installed and set with the required characteristic parameters to integrate the designed algorithm model. We manually insert an object into the simulation platform with a defined walking route around the particular location specified in the virtual environment part of the pipeline (Figure 1). The virtual simulation platform provides essential input frame sequences with feature information, such as ordinary, segmented, and grayscale (negative) depth images, that could proceed through tf-agent DRL network layers to learn and take action for target tracking measures. We use image depth to identify object location and targeted class while experimenting through the network for adaptation to unknown conditions.

B. TF-AGENT-BASED DRL OBJECT TRACKING MODEL

Tracking objects on a virtual simulation platform differs from typical state-of-the-art target-tracking framework approaches. The target object moves automatically across the simulation platform area, occluded by obstacles such as high walls, several different-shaped objects, etc. In this proposal, a random walking pedestrian was set into a simulation environment to create learning and tracking conditions by a virtual AirSim drone agent. As shown in Figure 2, we integrated the simulation platform with the suggested alternative algorithm model to jointly optimize representatives by experimenting in different conditions. Firstly, we request the environment simulation platform to get the typical depth images and the segmentation map to get the pixels with a target. In the next step, frames will be gray-scaled and normalized for further recognition of an object in the virtual simulation model. After getting the pixels with the target and creating them, bounding box points are concatenated and transformed into network-readable values for the following process.

1) DQN-BASED TF-AGENT

The DQN agent is suitable for any environmental condition possessed by a discrete action space formulated deterministically for simplicity and expectations over stochastic environmental transitions. The main goal of the DQN agent in this model is to train a policy to maximize the discounted cumulative reward (1).

$$\boldsymbol{R}_{t_0} = \sum\nolimits_{t=t_0}^{\infty} \gamma^{t-t_0} r_t \tag{1}$$

That is also known as the return value $\boldsymbol{R}_{t_0}$. In most RL-based networks, the discount factor γ should be a constant value highlighting the sum of converges between 0 and 1. It allows our agent to gain better reward value results by avoiding uncertain environment feature information and identifying which is less relevant than a fairly confident one. The $\boldsymbol{Q}^*$ is to achieve an affordable reward or return value emphasized: $\boldsymbol{Q}^* : \boldsymbol{State} \times \boldsymbol{Action} \longrightarrow \mathbb{R}$ when the action is taken in a given state, the return results from a constructed policy to achieve maximized rewards (2).

$$\pi^*(s) = \underset{a}{\mathbf{argmin}}\ Q^*(s, a) \tag{2}$$

In our virtual reality world simulation model, we will have access to state and action space information related to the $\boldsymbol{Q}^*$ value function to create and train the Q-network. Most of the $\boldsymbol{Q}$ functions in the case of policy-required conditions obey Bellman's [39] equation (3).

$$Q^{\pi}(s, a) = R + \gamma Q^{\pi}(s', \pi(s')) \tag{3}$$

The difference between initial and learned values calculations following the equality equation, also known as Q-value updating [39] or the temporal difference error (4).

$$\delta = Q^{new}(s,a) = (1-\alpha)\underbrace{Q(s,a)}_{old\ value} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} Q(s', a')\right)}^{learned\ value} \tag{4}$$

Equation (4) above calculates the updating Q-value for the state-action $(\boldsymbol{s}, \boldsymbol{a})$ pair at time step $\boldsymbol{t}$. It is assumed to be equal to a weighted sum of old and learned values, where the initial old value would be 0 since the agent is experiencing this particular state-action pair value. The old value is multiplied by $(\mathbf{1} - \boldsymbol{\alpha})$. α learning rate is denoted and set as $\boldsymbol{\alpha} = 0.001$ for our default training network. Instead of overwriting the newly calculated Q-value, the $\boldsymbol{\alpha}$ learning rate is set to determine the previously computed Q-value amount for the initial state-action pair. To retain the recently obtained Q-value later, we give a higher learning rate for the equal state-action match to adopt the drone agent quickly for the computed Q-value. However, it should be at a balanced learning rate to keep the trade-off between the previous and new Q-values for the

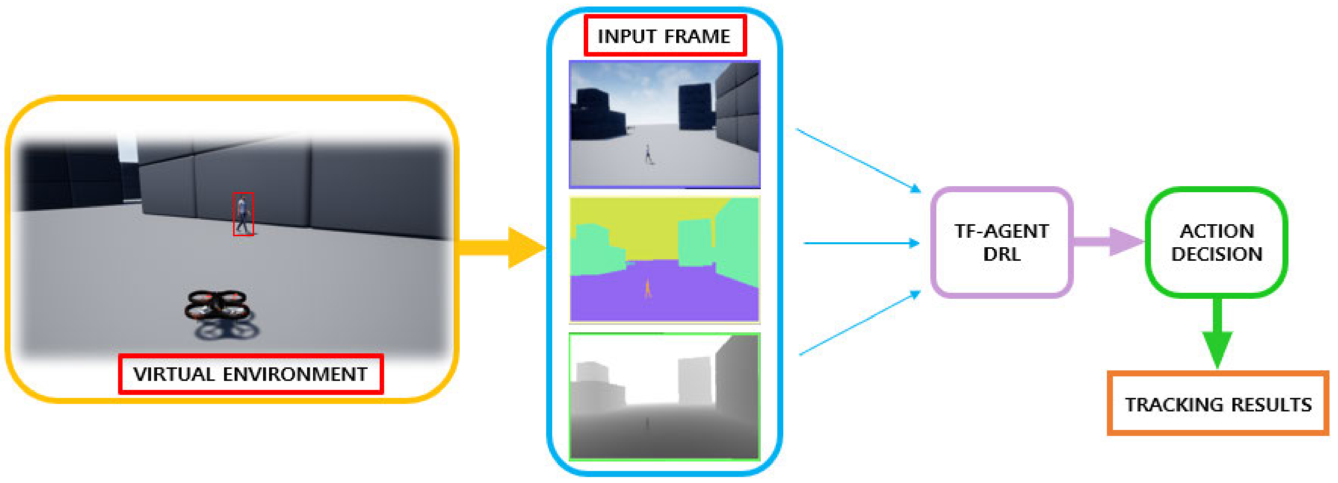


FIGURE 1. Proposed DRL-based TF-agent object tracking baseline integration with the Game Engine.

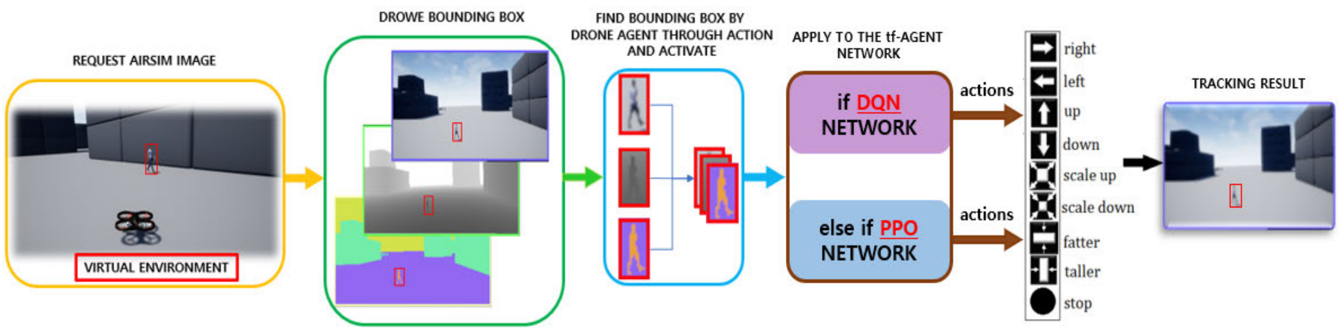


FIGURE 2. The flow chart of the DRL-based tf-agent object tracking model in Game Engine.

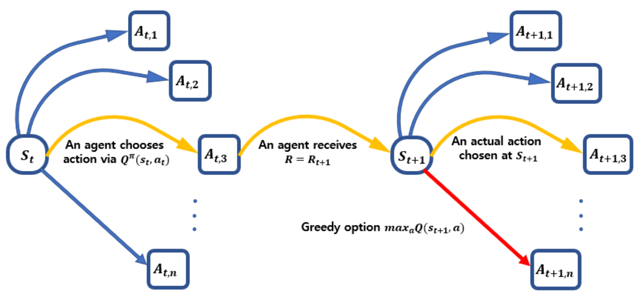

$A_{t,1}$
$A_{t,2}$
S_t
An agent chooses action via $Q^{\pi}(s_t, a_t)$
$A_{t,3}$
An agent receives $R = R_{t+1}$
S_{t+1}
An actual action chosen at S_{t+1}
$A_{t+1,3}$
$A_{t+1,1}$
$A_{t+1,2}$
Greedy option $max_a Q(s_{t+1}, a)$
$A_{t,n}$
$A_{t+1,n}$

FIGURE 3. The process of Q-learning update.

further training process. A learned value is a reward R_{t+1} that the drone agent receives moving randomly from the starting state point, plus discounted estimation γ of optimal future Q-value for a new state-action match $(s',a'$ n 1-time steps. The output of the learned value multiplication by the learning rate α is done to get the optimal policy value update. The Q-learning process update illustration is in Figure 3.

As illustrated in Figure 3, there can be several actions through the training or learning process where an agent chooses the seemingly optimal actions $Q^{\pi}(s_t,a_t)$ and receives a reward for the agent's performance through steps in a virtual environment. For further learning, the agent should choose an action from the S_{t+1} state to continuously learn and analyze the environment with more profound feature results. Here, the greedy epsilon option is a straightforward strategy for balancing exploration and exploitation by randomly selecting between the two. The method, where epsilon is the likelihood of selecting to explore or exploit, determines whether it proceeds to explore the environment with a slight chance. We can see the action selection with the epsilon greedy method mathematical formulation below the equation (5).

$$\text{Action at time (t) } a_t = \begin{cases} \max\limits_{a} Q_t(a) & 1-\epsilon \\ \textit{any action } (a) & \epsilon \end{cases} \tag{5}$$

The action selection method for further learning can be detailed thoroughly when the value is $1-\epsilon$, and the agent uses exploitation to take advantage of prior knowledge, which is a best-estimated reward; otherwise, ϵ it takes exploration to look for new optimal options.

The value of each action must be specified for our agent to choose the one that will result in the best reward. The action-value estimation function (6) uses probability theory to define these values. The predicted reward received when choosing an action from a list of all potential actions refer to the "value of that action". So, we utilize the "sample-average" approach to estimate the value of doing an action since the agent does not know the value of choosing a particular action.

$$Q_t(a) = \frac{\text{Sum of rewards when action } (a) \text{ taken before time } (t)}{\text{Number of times action } (a) \text{ taken before time } (t)} = \frac{\sum_{i=1}^{t-1} R_i}{t-1} \tag{6}$$

The agent will then select the action with the most outstanding estimated value, referred to as a greedy action, once the value $Q(s', a')$ has its pick rate.

2) PPO-BASED TF-AGENT

Proximal Policy Optimization (PPO) is a straightforward policy gradient approach for RL-based optimization problems that alternates between optimizing a "surrogate" objective function using stochastic gradient ascent and sampling data through interaction with the environment [30]. The main idea and difference between primary and novel policy gradient methods is that the multiple-update minibatch objective function is applied. At the same time, the standard model updates the gradient per data sample in a single epoch.

A policy gradient technique known as the Proximal Policy Optimization (PPO) algorithm is applied to improve the policy of a reinforcement learning agent. PPO is a set of algorithms that includes PPO1 and PPO2. In this proposal, we will focus on the PPO1 algorithm. The Clipped Surrogate Goal is a drop-in substitute for the policy gradient objective to increase training stability by restricting the policy change at each step. To address these and other difficulties, we may limit the amount, alter the policy, and ensure it constantly improves. Furthermore, implementing this model helps to integrate with a complete processing algorithm to achieve efficient samples from input images and minimize hyperparameter tuning indicators. It achieves the same performance improvements while avoiding complexity by optimizing the basics of the Clipped Surrogate Objective (7).

$$L_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$

$r_t(\theta)\hat{A}_t$ – identifies the same objective before, inside the minimization; $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ – this part of formulation is the same objective, but $r(\theta)$ is clipped between $(1-\epsilon, 1+\epsilon)$; The complete $min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$ – episode shows the min of the same objective from before and the clipped one;

The main objective of clipping surrogates is a region clipping process that prevents the algorithm from getting too greedy and trying to update too much at once while training and learning to leave the region with good samples for estimation and summarizing. PPO enables us to conduct many

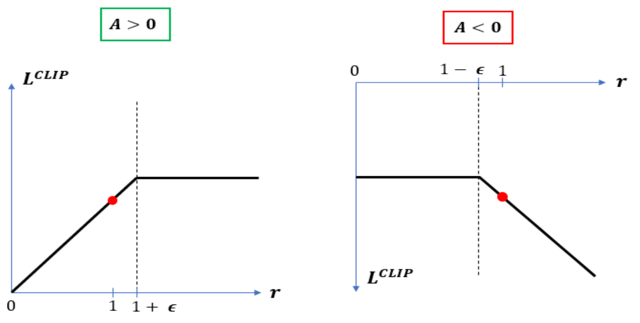


FIGURE 4. Probability ratio r of the surrogate function L^{CLIP} with positive $A > 0$ and negative $A < 0$ advantages. The red circle on each plot shows the starting point for the optimization, i.e., $r = 1$. The sum of the surrogate function L^{CLIP} can be performed for many terms [38].

gradient ascent epochs on our data stream without triggering harmfully massive policy modifications. Conducting these processes helps get more out of collected or streamed data while decreasing sample inefficiency. Moreover, the running PPO policy uses N parallel actors that individually collect data. The data is collected into mini-batches and then trained for K epochs using the Clipped Surrogate Objective function.

The Clipped Surrogate Objective will affect and optimize every action the agent takes. Updating should be stopped if the action is better (positive) $A > 0$ and more probable while taking the end of gradient steps. Otherwise, when the drone action is directed in the wrong direction but the action is good, it can be redirected or undone from the initial state. In the case of lousy action (negative) $A < 0$ and less probable outcomes gained from agents' actions, an agent needs to take short steps, or they do not need to go far steps in action space. When it comes to the normalized level of updating, it can be controlled in the balanced area to get the optimal probability ratio r for agents. The illustration of the L^{CLIP} surrogate function probability can be seen below in a summarizing Figure 4.

Figure 4, illustrates clipped surrogate objective functions optimization parameters for the running learning period with probability ratio. Effectively, this technique can be encouraged using significant policy changes across learning environments or input data for better probability optimization with a model agent. PPO with clipped object technique shows the difference between two maintained policy networks, the current $\pi_\theta(a_t|s_t)$ and the last used policy $\pi_{\theta_k}(a_t|s_t)$ applied to collect samples. A new policy evaluation comes from necessary sampling, which involves collecting old policy samples to improve efficiency.

The final loss function for the PPO actor-critic style looks below equation (8), a combination of the Clipped Surrogate Objective function, Value Loss Function, and Entropy bonus.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] \tag{8}$$

The given equation above includes several parts that can be complex to understand, yet it gives more priority to achieving more accurate results while applying them to the experimentation process. The explained first part is a Clipped Surrogate Objective function given in the (7) equation. In (8), given above, c_1 and c_2 are coefficients of the value of the related parameter for calculation. $L_t^{VF}(\theta)$ identifies as a squared-error value loss: $\left(V_\theta\left(s_t\right)-V_t^{targ}\right)^2$. To ensure the sufficient exploration of unknown and complex scenarios in virtual environments added an entry as a bonus $S\left[\pi_\theta\right]\left(s_t\right)$.

IV. EXPERIMENTAL RESULTS

One of the main objectives and focuses was to get the most advantage from the simulation platform to perform experiments in different conditions with parametric changes. Many related researchers used several simulation platforms to test and evaluate their algorithms in several evaluation studies. There are different methods to get an advantage from the realistic virtual platform. In most cases, platforms apply for experimenting purposes only. However, it can also be prioritized widely in learning, training, and testing. The current development of simulation platforms like Unity, Unreal Engine, and Cecium gives great opportunities and advantages to process and experiment with state-of-the-art models in multiple and impractical circumstances. One of the prime features of the simulators is the interconnection between programming languages (Javascript, Python, Go, Java, Kotlin, PHP, C#, Swift, etc.) and frameworks (Angular, jQuery, React, Ruby, and Rails, Vue, ASP.NET Core, Django, Express, etc.). However, building or setting up this type of architecture and framework is quite tricky, and it could only be successful in some cases due to third-party programs' and libraries' conflicts and disproportionality. In this research work, we conducted experiments with different parametric changes and finetuning, as explained in the following chapter sessions.

A. TRAINING RESULTS

We have trained our proposed model with a simple Bloks environment by inserting randomly moving objects to learn environmental space and to create a model for future testing and evaluation purposes. We applied two types of tf-agent models, DQN and PPO-based tf-agents, to achieve more comparable output results with a 0.001 learning rate configuration.

Figure 5 above illustrates the minimum reward outputs of the trained models in a typical Blocks environment, where the DQN-based tf-agent and the PPO-based tf-agent model are marked with blue and pink, respectively. The minimum reward is the smallest value that the agent can receive as a reward during the training process. The minimum reward is typically negative since most problems involve a penalty for making suboptimal decisions – the training epoch and reward at 2000 and 50, respectively. Furthermore, the background was set with a plot tab color in each method to show the overall performance of the training agents. Each agent model initially gained different rewards, whereas the DQN-based agent performed better. Nevertheless, at the end of the training epochs, the PPO-based agent receives better results than the DQN-based model agent. The whole training reward performance illustration in Figure 6 above, set to 2000 and 50, training epoch and reward, respectively. The maximum point is the highest value that the agent can receive in the training process. The maximum reward is typically a positive value since most problems involve a reward for making optimal decisions. The DQN-based TF-AGENT model initially gains a higher reward value in this graph. However, the PPO-based model performs better after 400 epochs until the end of the training steps. Understanding the range of possible rewards can help set the hyperparameters of the models, such as the learning rate or the discount factor. It can also help assess the performance of the trained agent, as the rewards obtained by

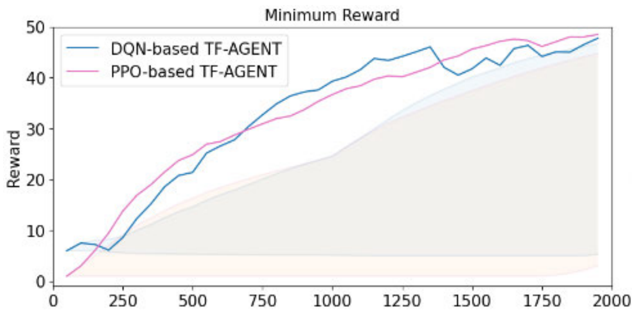


FIGURE 5. The minimum received reward output for two training models: DQN-based TF-AGENT and PPO-based TF-AGENT.

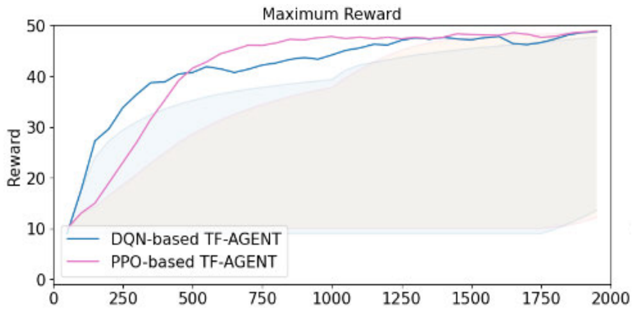


FIGURE 6. The received maximum reward output for two types of training models: DQN-based TF-AGENT and PPO-based TF-AGENT.

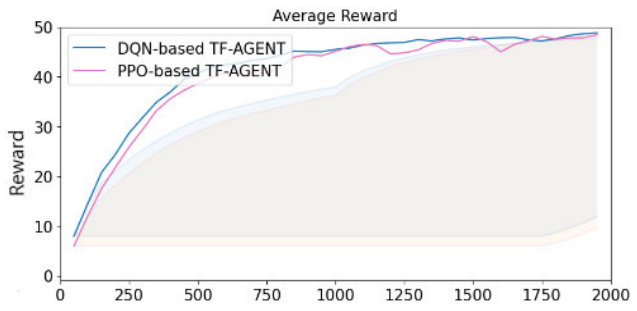
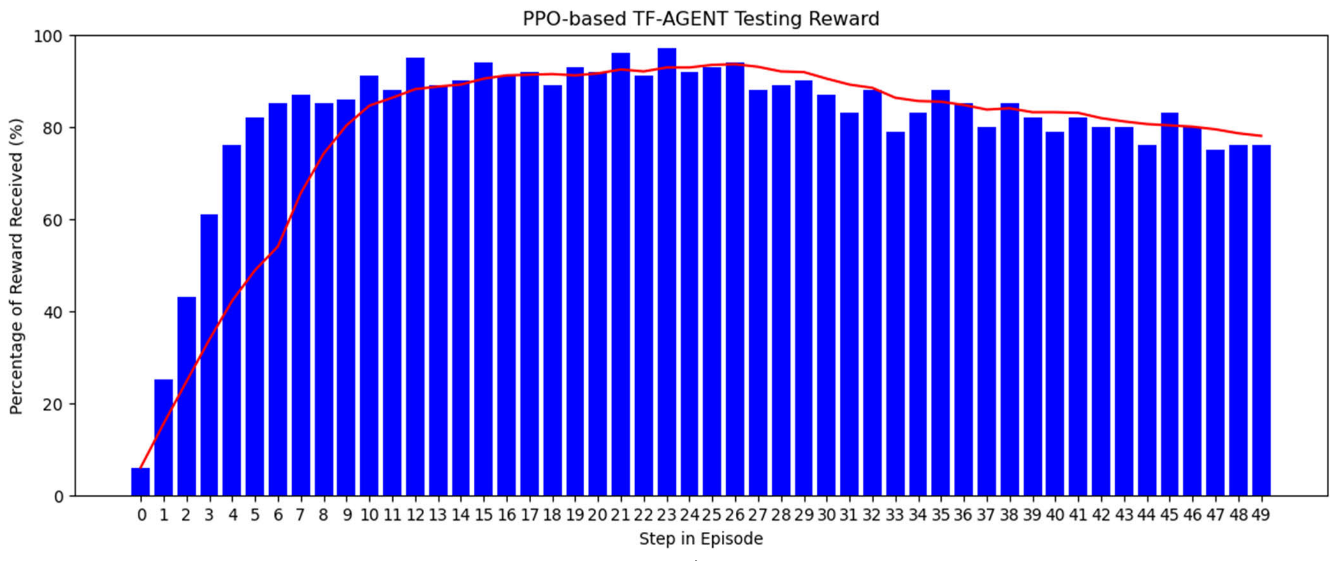
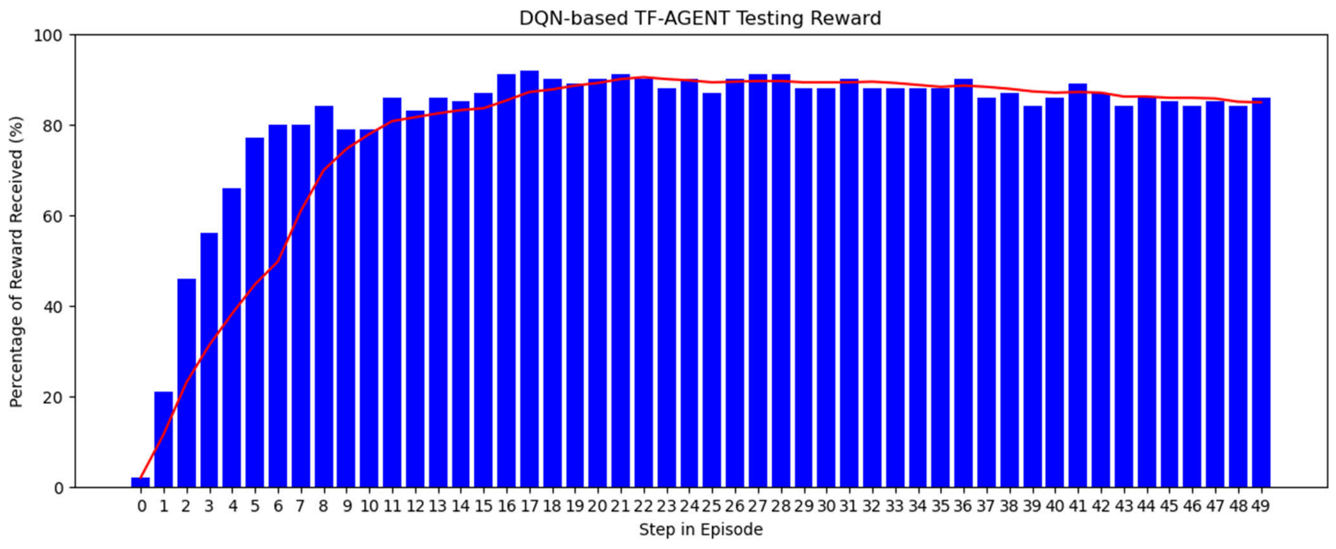


FIGURE 7. The received average rewards outcome of training for DQN and PPO-based TF-AGENTS.



a)



b)

FIGURE 8. The testing reward distribution of the DQN-based TF-AGENT (a) and the PPO-based TF-AGENT (b) in 50 steps of the episode.

the agent are compared against the minimum and maximum possible values.

The given average reward below refers to the mean value of the rewards received by agents during their interactions with the environment while training using DQN and PPO-based model algorithms. The average reward is essential for evaluating the agents' performance during training. During training, agents try to learn an optimal policy that maximizes the cumulative reward obtained over time. Calculating the average evaluation reward is done by dividing the sum of the rewards received during all episodes by dividing it by the total number of episodes. The estimated calculation is the average reward the agent will receive when interacting with the environment using the learned policy.

By evaluating the average reward received, we can see the difference between the DQN and PPO-based model's performance in varied configurations. However, in some scenarios, the average reward may not be the most suitable metric for evaluating the agent's performance.

B. TESTING RESULTS

We have tested our proposed DQN and PPO-based model agents with the same environmental condition but different unseen test episodes to explore the ability of the models and compare their performance. As mentioned above, the DRL-based algorithm's performance evaluation differs from other state-of-the-art algorithms in the case of performance metrics evaluations and comparison techniques. The agent-based models' precision can be seen or taken as a received reward value. As mentioned earlier, the average reward obtained by the agent during training can help create a model and apply this model to the testing process as a performance metric. This metric measures the agent's ability to navigate the environment and obtain expected output tracking. The diagram below (Figure 8) represents the DQN-based TF-AGENT model's output with the reward percentage received from unseen testing scenarios. In the testing session, the received reward percentage was set to 100 in the 50 steps respectively in every episode. The overall received in every step marked with a column and red line illustrates the smoothed value of the DQN-based TF-AGENT testing results trained and tested with standard reward in Figure 8 (a).

Figure 8 (b) shows the PPO-based TF-AGENT's received percentage reward testing results in the 50 steps of the episode, along with smoothed red line output. The different results between the DQN and PPO models received rewards in every training step. As we can see, the testing results show that both models give high accuracy and precise learning performance in every testing step output with an elevated conclusion.

V. CONCLUSION

In this research work, we have presented a DQN and PPO-based TF-AGENT model-based object tracking framework integrated with a simple Blocks environment to evaluate the performance of the proposed algorithm. It has been integrated with the simulation platform to highlight the algorithm's overall performance.

The simulation platform provides three types of essential input images to experiment with and evaluate the overall status. While testing in a virtual-reality scenario with virtual drone agents and finetuning to reach the best or desired results, the productivity and eligibility of these platforms are vital. The DQN and PPO-based virtual tf-agent drones learn how to detect and track an object inserted in this platform by obtaining consecutive frames from a primary Blocks environment and using a DRL network to manage the actions, states, and tracking pipeline. Both tf-agents are trained in a Blocks environment to adapt to the surroundings and existing objects in a simulation condition for additional testing, tracking accuracy, and speed assessment. In the training process, both models showed presentable results: minimum 49 (PPO) and 48 (DQN) rewards in 2000 epochs; maximum 49 (DQN) and 49 (PPO) rewards in 2000 epochs; average 49 rewards were received for both (PPO and DQN) models. Models performance contrasted 50 steps of one episode testing set, where the PPO-based tf-agent gets its pick value reward of 97% in step 23, DQN-based agent receives its max value of 86% in the 17$^{\text{th}}$ step respectively. However, the overall performance of the received percentage reward graph (Figure 8, a and b) indicates that the DQN-based model sequent performs better than the PPO-based one. Regarding stability, reward contribution, and numeric graphical performance, we examined and compared the algorithm techniques to various established hyperparametric changes with reinforcement learning-based network control incorporated into the simulation process. In future work, we are going to integrate our model with several state-of-the-art tracking techniques to improve the performance of the target tracking framework by testing it in more complex virtual simulation environments.

REFERENCES

[1] The Manufacturer. (Jun. 29, 2022). *The Benefits of Drones in Manufacturing*. [Online]. Available: https://www.themanufacturer.com/articles/the-benefits-of-drones-in-manufacturing/

[2] Croptracker. (Apr. 26, 2022). *Drone Technology in Agriculture*. Dragonfly IT. [Online]. Available: https://www.croptracker.com/blog/drone-technology-in-agriculture.html

[3] G. McNeal. (Nov. 2014). *Drones and Aerial Surveillance: Considerations for Legislatures*. [Online]. Available: https://www.brookings.edu/research/drones-and-aerial-surveillance-considerations-for-legislatures/

[4] B. Purahong, T. Anuwongpinit, A. Juhong, I. Kanjanasurat, and C. Pintaviooj, "Medical drone managing system for automated external defibrillator delivery service," *Drones*, vol. 6, no. 4, p. 93, Apr. 2022, doi: 10.3390/drones6040093.

[5] N. Tuśnio and W. Wróblewski, "The efficiency of drones usage for safety and rescue operations in an open area: A case from Poland," *Sustainability*, vol. 14, no. 1, p. 327, Dec. 2021, doi: 10.3390/su14010327.

[6] M. Elloumi, R. Dhaou, B. Escrig, H. Idoudi, and L. A. Saidane, "Monitoring road traffic with a UAV-based system," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6, doi: 10.1109/WCNC.2018.8377077.

[7] A. Chodorek, R. R. Chodorek, and A. Yastrebov, "Weather sensing in an urban environment with the use of a UAV and WebRTC-based platform: A pilot study," *Sensors*, vol. 21, no. 21, p. 7113, Oct. 2021, doi: 10.3390/s21217113.

[8] K. Jewani, M. Katra, D. Motwani, and G. Jethwani, "Firefighter drone," in *Proc. 1st Int. Conf. Adv. Sci. Innov. Sci., Eng., Technol. (ICASISET)*, 2020.

[9] C. Huang, C.-E. Lin, Z. Yang, Y. Kong, P. Chen, X. Yang, and K.-T. Cheng, "Learning to film from professional human motion videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4239–4248, doi: 10.1109/CVPR.2019.00437.

[10] A. A. El-Shafie, M. Zaki, and S. E. D. Habib, "Fast CNN-based object tracking using localization layers and deep features interpolation," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1476–1481, doi: 10.1109/IWCMC.2019.8766466.

[11] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object detection and tracking, based on DNN, for autonomous vehicles: A review," *IEEE Sensors J.*, vol. 21, no. 5, pp. 5668–5677, Mar. 2021, doi: 10.1109/JSEN.2020.3041615.

[12] X. Farhodov, K.-S. Moon, S.-H. Lee, and K.-R. Kwon, "LSTM network with tracking association for multi-object tracking," *J. Korea Multimedia Soc.*, vol. 23, no. 10, pp. 1236–1249, Oct. 2020.

[13] D. Gözen and S. Ozer, "Visual object tracking in drone images with deep reinforcement learning," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 10082–10089.

[14] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," 2017, *arXiv:1705.05065*.

[15] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.

[16] G. Yang, "Asymptotic tracking with novel integral robust schemes for mismatched uncertain nonlinear systems," *Int. J. Robust Nonlinear Control*, vol. 33, no. 3, pp. 1988–2002, Feb. 2023.

[17] G. Yang, T. Zhu, F. Yang, L. Cui, and H. Wang, "Output feedback adaptive RISE control for uncertain nonlinear systems," *Asian J. Control*, vol. 25, no. 1, pp. 433–442, Jan. 2023.

[18] M. Bhattarai and M. Martínez-Ramón, "A deep Q-learning based path planning and navigation system for firefighting environments," in *Proc. 13th Int. Conf. Agents Artif. Intell. (ICAART)*, 2021, pp. 267–277, doi: 10.5220/0010267102670277.

[19] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, and S. Sakri, "Deep reinforcement learning for vision-based navigation of UAVs in avoiding stationary and mobile obstacles," *Drones*, vol. 7, no. 4, p. 245, Apr. 2023, doi: 10.3390/drones7040245.

[20] W. Zhao, Z. Meng, K. Wang, J. Zhang, and S. Lu, "Hierarchical active tracking control for UAVs via deep reinforcement learning," *Appl. Sci.*, vol. 11, no. 22, p. 10595, Nov. 2021, doi: 10.3390/app112210595.

[21] Z. Wang, H. Li, Z. Wu, and H. Wu, "A pretrained proximal policy optimization algorithm with reward shaping for aircraft guidance to a moving destination in three-dimensional continuous space," *Int. J. Adv. Robotic Syst.*, vol. 18, no. 1, Jan. 2021, Art. no. 172988142198954, doi: 10.1177/1729881421989546.

[22] Z. Tan and M. Karaköse, "A new approach for drone tracking with drone using proximal policy optimization based distributed deep reinforcement learning," *SoftwareX*, vol. 23, Jul. 2023, Art. no. 101497, doi: 10.1016/j.softx.2023.101497.

[23] M. A. B. Abbass and H.-S. Kang, "Drone elevation control based on Python-unity integrated framework for reinforcement learning applications," *Drones*, vol. 7, no. 4, p. 225, Mar. 2023, doi: 10.3390/drones7040225.

[24] E. Çetin, C. Barrado, and E. Pastor, "Countering a drone in a 3D space: Analyzing deep reinforcement learning methods," *Sensors*, vol. 22, no. 22, p. 8863, Nov. 2022, doi: 10.3390/s22228863.

[25] J.-H. Park, K. Farkhodov, S.-H. Lee, and K.-R. Kwon, "Deep reinforcement learning-based DQN agent algorithm for visual object tracking in a virtual environmental simulation," *Appl. Sci.*, vol. 12, no. 7, p. 3220, Mar. 2022, doi: 10.3390/app12073220.

[26] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2488–2496.

[27] B. Li and Y. Wu, "Path planning for UAV ground target tracking via deep reinforcement learning," *IEEE Access*, vol. 8, pp. 29064–29074, 2020, doi: 10.1109/ACCESS.2020.2971780.

[28] C.-C. Chang, J. Tsai, P.-C. Lu, and C.-A. Lai, "Accuracy improvement of autonomous straight take-off, flying forward, and landing of a drone with deep reinforcement learning," *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 914–919, 2020.

[29] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1317–1332, Jun. 2020.

[30] D. Zhang, Z. Zheng, R. Jia, and M. Li, "Visual tracking via hierarchical deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 4, pp. 3315–3323.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[32] J. Supancic and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 322–331.

[33] W. Wang, Y. Huang, and L. Wang, "Language-driven temporal activity localization: A semantic matching reinforcement learning model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 334–343.

[34] W. Wu, D. He, X. Tan, S. Chen, and S. Wen, "Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6221–6230.

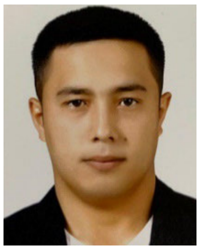
[35] J. Han, L. Yang, D. Zhang, X. Chang, and X. Liang, "Reinforcement cutting-agent learning for video object segmentation," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9080–9089.

[36] K. Ko, "Visual object tracking for UAVs using deep reinforcement learning," ProQuest dissertation, Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, 2020, doi: 10.31274/etd-20200624-169.

[37] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, Nov. 2018.



KHURSHEDJON FARKHODOV received the B.S. degree from the Department of Computer Engineering, Tashkent University of Information Technologies, Uzbekistan, in 2017, and the M.S. degree from the Department of AI Convergence and Application Engineering, Pukyong National University, South Korea, in 2021, where he is currently pursuing the Ph.D. degree. His research interests include digital image processing, computer vision, and machine learning.

SUK-HWAN LEE received the B.S., M.S., and Ph.D. degrees in electrical engineering from Kyungpook National University, South Korea, in 1999, 2001, and 2004 respectively. He is currently a Professor with the Department of Computer Engineering, Dong-A University. His research interests include multimedia security, digital image processing, and computer graphics.

JAN PLATOS (Member, IEEE) received the Ph.D. degree in computer science, in 2010. He became a Full Professor with the Department of Computer Science, in 2021. Since 2021, he has been the Dean of the Faculty of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava. He has coauthored more than 240 scientific papers published in conference proceedings and journals. His citation report consists of 849 citations and an H-index of 13 on the Web of Science, 1407 citations and an H-index of 16 on Scopus, and 2078 citations and an H-index of 21 on Google Scholar. His research interests include machine learning, artificial intelligence, industrial data processing, text processing, data compression, bioinspired algorithms, information retrieval, data mining, data structures, and data prediction.

KI-RYONG KWON received the B.S., M.S., and Ph.D. degrees in electronics engineering from Kyungpook National University, in 1986, 1990, and 1994, respectively. He was with Hyundai Motor Company, from 1986 to 1988, and the Pusan University of Foreign Language, from 1996 to 2006. He was a Postdoctoral Researcher with the University of Minnesota, USA, from 2000 to 2002, a Visiting Professor with Colorado State University, from 2011 to 2012, and the General President of the Korea Multimedia Society, from 2015 to 2016. He is currently a Professor with the Department of IT Convergence and Application Engineering, Pukyong National University. His research interests include digital image processing, multimedia security and watermarking, bioinformatics, and weather radar information processing. He is the Director of the IEEE R10 Changwon Section.